INSTITUTUL POLITEHNIC "TRAIAN VUIA" TIMISOARA

FACULTATEA DE ELECTROTEHNICA

ing.OLIMPIU NEGRU

CONTRIBUTII PRIVIND METODELE DE TESTARE A PLACHETELOR ECHIPATE CU CIRCUITE INTEGRATE LOGICE

- Teză de doctorat -

Conducător ştiinţific:
Prof.dr.ing.ALEXANDRU ROGOJAN

- 1982 -

C U P R I N S

Cap.I. INTRODUCERE

Produsele realizate în actuala etapă de dezvoltare sînt caracterizate printr-un raport complexitate/dimensiuni de gabarit în continuă creştere [H2] , [L1] .

Pe măsura amplificării complexităţii şi performanţelor produselor, problema siguranţei în exploatare devine tot mai acută, astfel încît evaluarea obiectivă a indicilor de calitate apare tot mai presantă, dar şi mai greu de realizat. Asigurarea calităţii, ca o totalitate a trăsăturilor produsului ce-l fac capabil să satisfacă o necesitate dată, implică o integrare a aspectelor de proiectare, producţie, exploatare şi întreţinere [K2] . In acest caz, în toate fazele de elaborare şi exploatare, operaţia de testare apare ca o parte integrantă a proceselor de fabricaţie şi exploatare, cu o pondere crescîndă în ceea ce priveşte timpul şi costul aferent, motiv pentru care automatizarea ei se impune ca o necesitate.

Un experiment de testare este procesul prin care se aplică o secvenţă de semnale corespunzătoare, pe intrările obiectului ce se verifică şi, plecînd de la o stare cunoscută, se efectuează o analiză asupra răspunsului. Stabilirea corectitudinii răspunsului se face prin compararea acestuia cu un semnal corect, obţinut de la un obiect martor sau sintetizat pe baza funcţiei ce stabileşte corespondenţa dintre intrările, stările şi ieşirile obiectului.

Testarea se poate desfăşura sub forma unui experiment programat, cînd întreaga secvenţă de semnale, aplicate pe intrările obiectului care se verifică, se determină anticipat sau ca un experiment adaptiv, în care fiecare tip de semnal, în afară de primul, este stabilit pe baza observaţiilor şi rezultatelor obţinute anterior [A4] . In al doilea caz există posibilitatea ca faza de localizare a defectelor să ducă la rezultate superioare.

Secvenţele de semnale se generează pe baza unui algoritm, astfel conceput încît rezultatul să fie obţinut rapid şi sigur. Intrucît orice obiect are un număr limitat de intrări, respectiv ieşiri, eficacitatea testării depinde atît de algoritm, cît şi

de modul în care este construit obiectul, respectiv accesibil. Prin această ultimă observaţie se ajunge la concluzia că obiectelor trebuie să li se atribuie, în plus, o proprietate de testabilitate, prin care fiecare element component să poată fi controlat de la cel puţin o intrare, iar comportarea lui, vizualizată la cel puţin o ieşire [*14], [*9], [F1].

Integrarea pe scară tot mai largă, alături de realizarea modulară, bazată pe elemente cu caracter universal, reduc, simţitor, ciclul de elaborare al unui produs final. Dar în acelaşi timp, creşte ponderea etapelor de punere în funcţiune, aferente fiecărei faze de elaborare. Dacă dimensiunile de gabarit ale fiecărui element component, începînd cu circuitul integrat, apoi placheta, subansamblul, deci şi produsul final, se reduc, aceasta are ca efect îngreunarea accesului la diferite componente, complicînd, într-o măsură tot mai mare, testarea şi evaluarea parametrilor.

Toate aceste consideraţii concură la necesitatea asigurării unei strategii de testare, care să rezolve problemele caracteristice fiecărei faze de elaborare a unui produs [C4]. Volumul şi profilul producţiei influenţează strategia de testare prin mărimea şi complexitatea seriilor de produse fabricate într-o perioadă calendaristică dată [A5]. In acest caz este necesară realizarea următoarelor nivele de testare :

- testarea componentelor (circuite integrate, tranzistoare, diode, rezistenţe, condensatoare);
- testarea plachetelor neechipate;
- testarea plachetelor echipate;
- testarea sertarelor şi a cablajelor;
- testarea şi evaluarea parametrilor produsului final.

După modul de conducere al operaţiilor de testare, aceasta poate fi :

- manuală;
- semiautomată;
- automată;
- dirijată cu calculator.

Intr-un experiment de testare, se pot realiza următoarele tipuri de verificări : parametrice, funcţionale, dinamice sau combinaţii ale acestora.

O modalitate aparte o constituie testarea "IN CIRCUIT", tehnică ce permite, în urma accesului direct pe placheta, cu a-

jutorul unui pat de cuie, efectuarea testării parametrice şi funcţionale a fiecărei componente [ ] , [C4] .

Testarea componentelor integrate şi discrete, la producător, cuprinde toate tipurile de verificări. Dar aceasta se face numai pentru un set de mostre dintr-un lot. Decizia de acceptare sau respingere a întregului lot este luată în funcţie de numărul defectelor depistate în setul de mostre. Deci se vor găsi componente defecte şi într-un lot controlat. Efectul acceptării loturilor, fără o testare prealabilă a fiecărei componente, se prezintă în fig.1.1 [N3] . Se observă că dacă 2% din componente sînt defecte, pe o plachetă cu 5o de componente, doar 35 din loo plachete au şansa să nu fie defecte.

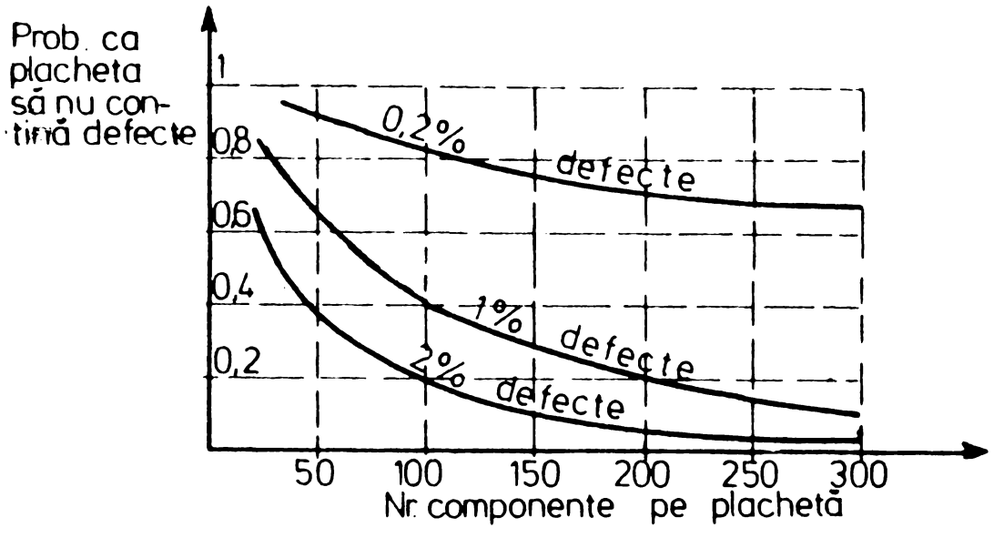


Fig.1.1.

Dacă se consideră şi faptul că detectarea şi localizarea unei componente defecte, pe o plachetă echipată, este o operaţie mult mai dificilă decît testarea individuală a componentei, rezultă importanţa asigurării unei tehnologii optime de testare a componentelor, la beneficiar.

Testarea plachetelor neechipate urmăreşte verificarea continuităţii traseelor şi detectarea scurtcircuitelor dintre ele şi se realizează cu echipamente specializate.

Testarea plachetelor echipate este una dintre cele mai dificile faze ale tehnologiei de testare.

In funcţie de numărul şi complexitatea componentelor cu care sînt echipate, plachetele se pot clasifica în următoarele tipuri :

a) plachete echipate numai cu circuite integrate SSI, în

număr de 3o-1oo circuite. Este cazul plachetelor mai puţin complexe, proiectate şi executate în perioada 196o-197o, în momentul de faţă fabricîndu-se, în cea mai mare parte, ca piese de schimb. Plachetele sînt realizate cu un singur strat, iar accesul se asigură printr-un singur conector;

b) plachetele ce conţin 5o-3oo circuite integrate SSI şi MSI sînt, în momentul de faţă, cele mai răspîndite. Accesul pe plachetă se realizează cu ajutorul a 1-2 conectoare, în cazuri de excepţie putînd fi chiar 3-4;

c) plachetele ce conţin 4o-15o circuite logice, alături de unele circuite analogice. Pentru acces utilizează 1-2 conectoare;

d) plachete echipate cu 1o-1oo circuite SSI, MSI şi LSI.

Chiar dacă numărul de circuite LSI este redus, funcţionarea plachetei este complexă şi, implicit, dificil de verificat. Accesul pe plachetă se realizează cu ajutorul a 1-4 conectoare.

Plachetele caracteristice tipurilor b, c, d conţin 1-4 straturi.

Pe fiecare din tipurile prezentate mai sus pot apare, în cazuri de excepţie şi în număr redus, componente semiconductoare utilizate pentru adaptări de nivele, iniţializări etc.

In general, tendinţa este de a utiliza plachete, ale căror dimensiuni să permită echiparea cu un număr tot mai mare de circuite de orice tip, astfel încît să se reducă numărul total de plachete ale produsului, iar cu aceasta şi interconexiunile dintre ele [*14].

Din analiza tipurilor de plachete echipate se pot evidenţia următoarele elemente caracteristice :

- creşterea continuă a complexităţii schemelor electronice realizate pe plachete;
- scăderea continuă a raportului puncte de acces/număr de componente de pe o plachetă;
- existenţa, pe plachete, a unui număr redus de componente semiconductoare, solicitate de modul de funcţionare al unor circuite, adaptări, protecţii etc.;
- circuitele integrate cu care se echipează placheta pot fi realizate în tehnologii diferite, ca : bipolară, MOS, CMOS, ECL etc., ceea ce poate determina existenţa de nivele logice diferite, chiar pe aceeaşi plachetă;
- necesitatea alimentării unor plachete de la mai multe surse;

- caracterul dinamic al unor circuite integrate LSI implică condiţionări suplimentare, realizate din exterior, necesare în cazul studiului funcţionării la nivel de plachetă echipată;
- existenţa unor stări ale plachetei, care nu pot fi controlate sau vizualizate direct prin conector.

Toate aceste elemente influenţează negativ faza de testare a plachetelor echipate, contribuind la evidenţierea acesteia ca una dintre cele mai dificile din cadrul fluxului tehnologic de testare, aferent procesului de elaborare al produsului.

Testarea plachetelor echipate, în condiţii reale de funcţionare, solicită o investiţie importantă de timp şi mijloace şi se poate realiza, într-un mod adecvat, cu ajutorul echipamentelor de testare conduse prin calculator şi cunoscute sub denumirea de sisteme de testare automată.

Complexitatea acestei faze de testare impune respectarea unei anumite metodologii, cu avantaje, atît în facilitatea operaţiei de testare propriu-zisă, cît şi în posibilitatea efectuării acesteia cu ajutorul unor sisteme de testare mai simple.

In general, distribuţia defectelor pe plachetele echipate este următoarea [A5] :

- 4o-45% - defecte de fabricaţie, cum sînt : lipituri reci, scurtcircuite, lipsă lipituri etc.;
- 3o-35% - defecte datorită componentelor, ca : componente defecte, componente care nu se încadrează în clasă, erori de marcare a componentelor, componente greşit utilizate etc.;
- 22-26% - defecte de asamblare : componente greşit poziţionate, polarităţi incorecte etc.

Distribuţia defectelor este în funcţie de complexitatea şi eficienţa procesului de producţie. In fig.1.2 se prezintă : a) distribuţiile de defecte tipice, b) implicaţiile fiecărui tip de defect asupra timpului solicitat pentru detectare [*18], [A5].

In special la seriile mici, procentajul de plachete defecte din cauza scurtcircuitelor este mare, datorită lipsei de timp necesar optimizării funcţionării băii de lipit cu undă staţionară. Se apreciază că din 7 defecte, maximum 4 pot fi scurtcircuite [A5]. Din acest motiv, testarea va urmări, în primul rînd, detectarea scurtcircuitelor, operaţie ce se poate efectua cu un echipament specializat. Astfel se asigură şi o creştere importan-

tă a capacităţii şi disponibilităţii sistemului de testare complex.

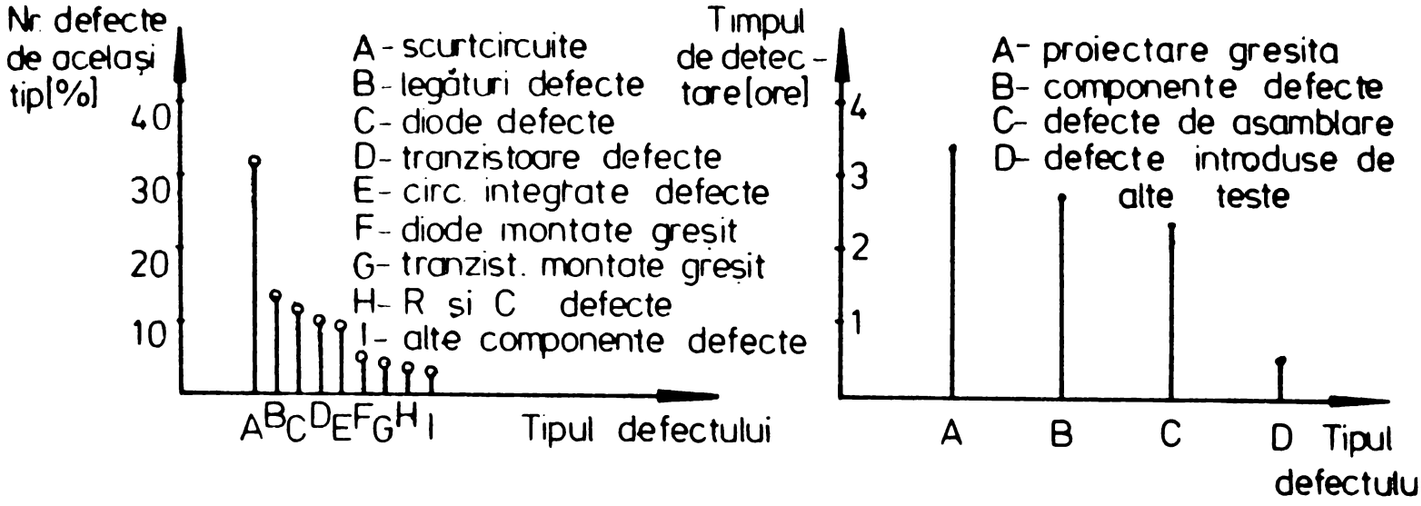


Fig.1.2.

Dacă se utilizează tehnica de testare "In Circuit", se poate realiza atît verificarea continuităţii (scurtcircuite, întreruperi, lipsă legături), cît şi testarea parametrică a componentelor cu care este echipată placheta. In plus, programele de testare sînt relativ simple. Dezavantajul major este că solicită o interfaţă cu pat de cuie, între sistemul de testare şi placheta sub test, interfaţă scumpă şi greu de realizat.

Dacă se efectuează o testare funcţional-dinamică, accesul pe plachetă se face prin conector. Deci dispare patul de cuie, dar cu aceasta şi posibilităţile de acces direct pe plachetă, iar programele de testare devin mai complexe.

Cele două tehnici nu se exclud ci se completează reciproc. Testarea "In Circuit" detectează defectele permanente, ce afectează, în special, comportarea individuală a componentelor. Testarea funcţional-dinamică este mai completă, permiţînd detectarea şi a defectelor aleatoare, a hazardului etc.

In fig.1.3 se prezintă o analiză comparativă a eficienţei utilizării diferitelor modalităţi de testare a plachetelor echipate şi anume [*18], [*19] :

(1) - testarea cu un echipament simplu (eventual manual, plus instrumente de măsură universale) şi personal special calificat;

(2) - testarea cu un echipament ce utilizează plachete martor;

(3) - testarea cu ajutorul unui sistem cu calculator, care

utilizează plachete etalon;

④ - testarea cu ajutorul unui sistem cu calculator, care utilizează tehnica răspunsului sintetizat şi elaborarea automată a programelor de testare.

Se observă că, începînd cu un volum de 5ooo plachete/an, cea mai avantajoasă modalitate de testare este evidenţiată de diagrama ④.

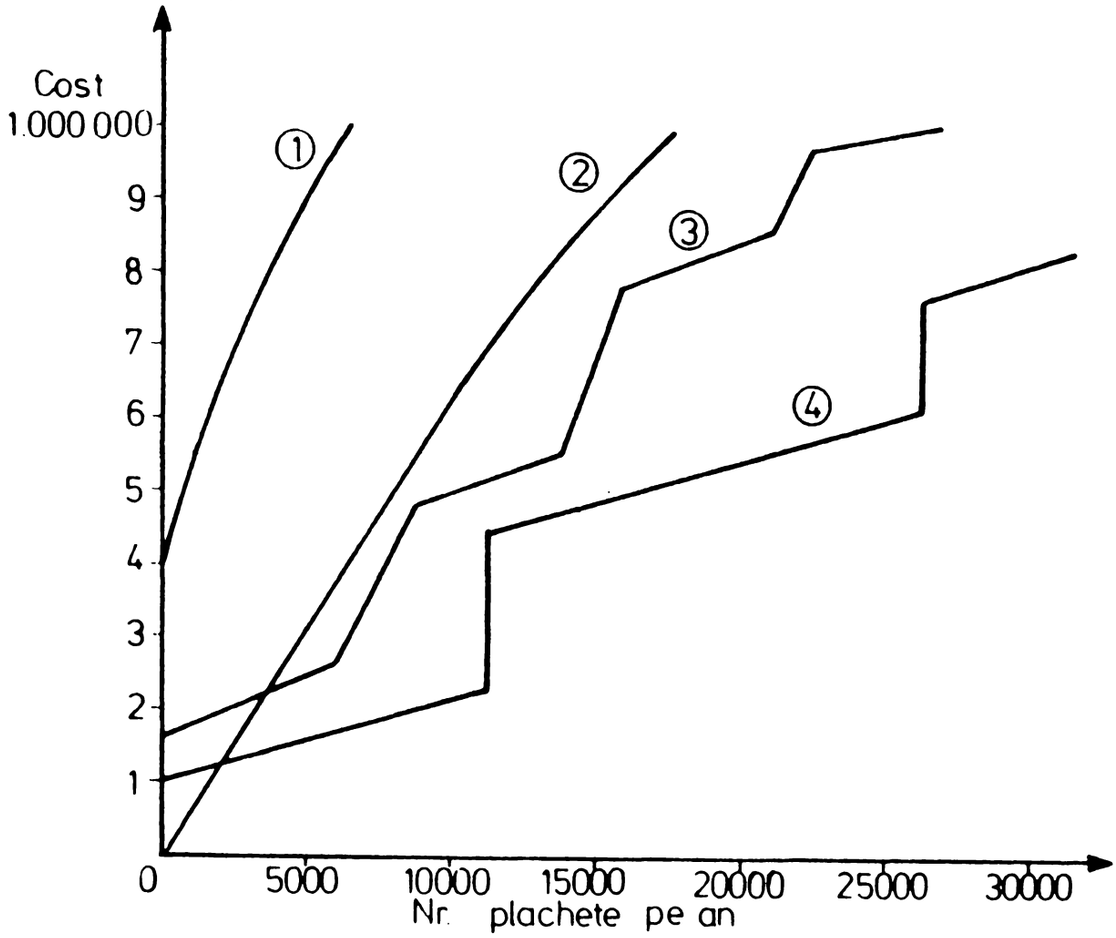


Fig.1.3.

Testarea sertarelor se face cu echipamente specializate şi urmăreşte verificarea legăturilor între două sau mai multe puncte. Dacă numărul punctelor este mare, creşte numărul capetelor de acces. Căutarea legăturilor, pe porţiuni de sertar, implică memorarea unei cantităţi de informaţie, care se amplifică exponenţial cu avansul spre final, dar reduce numărul capetelor de acces. Din aceste motive, modalitatea de lucru se va stabili în funcţie de situaţia reală [*37].

Testarea produselor se realizează cu echipamente specializate, urmărindu-se verificarea diferitelor moduri de lucru, proceduri, evaluarea performanţelor etc. Astfel de echipamente pot fi simulatoarele de parametri şi/sau de procese, care asigură

evaluarea parametrilor produsului analizat, în condiţii reale de funcţionare.

Din analiza diferitelor strategii de testare rezultă următoarele concluzii [C4] :

- în majoritatea cazurilor, testarea continuităţii, urmată de cea "In Circuit" şi apoi de cea funcţional-dinamică determină strategia optimă;
- în toate cazurile, testarea componentelor şi a plachetelor neechipate scade, substanţial, costul testării şi al reparării plachetelor echipate;
- dacă nu s-a făcut o testare a componentelor, testarea "In Circuit" este mai indicată decît cea funcţional-dinamică, exceptînd cazurile cînd defectele funcţionale sînt în număr mai mare decît 5o% din totalul defectelor;
- dacă defectele de continuitate reprezintă peste 5o% din totalul defectelor, o strategie optimă este acea care asigură o testare a continuităţii pentru plachetele echipate, urmată de o testare funcţional-dinamică.

Eficacitatea operaţiei de testare este direct influenţată de metodele utilizate pentru întocmirea programelor de testare. Se programează diferite metode cu ajutorul cărora, plecînd de la schema electronică realizată pe plachetă, se construiesc secvenţele de semnale ce se aplică pe pinii conectoarelor plachetei, stabiliţi ca intrări, precum şi răspunsurile aşteptate pe pinii conectoarelor, stabiliţi ca ieşiri. Prin testare se urmăreşte detectarea defectelor la nivel de pin al conectorului şi dacă este posibil, localizarea lor în interiorul plachetei.

In momentul de faţă nu există o metodă general valabilă pentru testarea oricărui tip de schemă logică, realizată pe o plachetă. S-au elaborat o serie de metode destinate, în general, fie pentru tipul de scheme combinaţionale, fie pentru cele secvenţiale. Chiar şi în cadrul aceluiaşi tip există numeroase cazuri cînd aplicabilitatea unei anumite metode este mai restrînsă. Există o clasă de scheme logice, la care se poate aplica optim o anumită metodă. Problemele se complică şi mai mult dacă schema logică conţine circuite LSI [H2], [K6], [H9], [J1].

Cele mai cunoscute metode destinate testării schemelor logice combinaţionale sînt : metoda tabelei de adevăr, metoda diferenţelor booleene, metoda Poage, algoritmul D, metoda Armstrong [C3], [F2], [L2], [W1].

Metoda tabelei de adevăr construieşte tabelele de adevăr pentru circuitul corect şi cel defect. Vectorii de testare destinaţi detectării unui anumit defect se obţin comparînd cele două tabele. Pe baza acestor tabele se poate construi setul minim al vectorilor de testare pentru o schemă logică. Deficienţa majoră a acestei metode este determinată de dimensiunea mare a tabelelor, chiar şi pentru schemele logice simple.

Metoda diferenţelor booleene urmăreşte realizarea operaţiei SAU-EXCLUSIV între două funcţii booleene, una descriind schema logică corectă iar cealaltă pentru schema logică defectă. Dacă diferenţa booleană pentru un anumit defect este 1, acel defect este detectabil. Metoda este indicată pentru schemele logice simple, în cazul celor mai complexe, volumul de calcule creşte foarte mult. Totodată permite abordarea şi a defectelor multiple, dar, bineînţeles, cu un mare volum de calcule.

Algoritmul D este una dintre metodele cele mai utilizate pentru detectarea defectelor în schemele logice combinaţionale. Acesta urmăreşte construirea vectorilor de testare pentru fiecare defect în parte, ce ar putea apare în schema logică. Diferenţa de bază faţă de celelalte metode este că nu solicită descrierea şi memorarea întregii scheme logice sub formă de tabele sau formule. Dar pentru schemele logice complexe, cu un număr mare de nivele de la locul apariţiei defectului pînă la o ieşire accesibilă, stabilirea vectorilor de testare solicită un volum mare de calcule şi capacitate mare de memorare.

Spre deosebire de metodele prezentate, metoda Poage permite determinarea setului complet al vectorilor de testare. Metoda se bazează pe descrierea funcţională a fiecărei ieşiri a schemei logice. Fiecare expresie a unei ieşiri nu reprezintă numai relaţia dintre variabilele ce determină valoarea respectivei ieşiri, ci reţine şi efectul oricărui defect ce ar afecta valoarea uneia din variabilele ce influenţează ieşirea dată. Metoda este indicată numai pentru cazul schemelor logice relativ simple.

Renunţînd la unele informaţii privind, în special tipul defectelor, Armstrong elaborează o altă metodă, asemănătoare cu precedenta dar mai simplă. Totodată, pentru cazul schemelor logice mai simple, poate fi aplicată şi în vederea detectării defectelor multiple.

In cazul schemelor logice secvenţiale, determinarea vectorilor de testare este o problemă mult mai dificilă, datorită par-

ticularităţilor ce apar în funcţionarea circuitelor secvenţiale [C3],[F23],[I2],[W1],[Z1],[Z2]. Se utilizează, fie unele din metodele prezentate în cazul schemelor combinaţionale, fie metode proprii schemelor secvenţiale. Din prima categorie, rezultatele cele mai bune s-au obţinut cu ajutorul algoritmului D extins. Pentru a putea fi aplicat, fiecare circuit secvenţial se tratează ca o mulţime de circuite combinaţionale.

Dintre metodele specifice, cele mai utilizate sînt cele, care se bazează pe identificarea maşinii (metoda Hennie, metoda Hsieh etc.). Modul de lucru urmăreşte determinarea vectorului de testare, care asigură ca tabela de tranziţii a schemei logice corecte să fie diferită de toate celelalte tabele de tranziţii ale schemelor logice ce conţin defecte. Faptul că se cere construirea unor tabele de tranziţii este un dezavantaj, acesta fiind destul de dificil de realizat, mai ales în cazul schemelor complexe. Probleme deosebite pot apare în cazul realizării experimentelor de iniţializare, necesare pentru aducerea schemei într-o stare cunoscută.

Ţinînd cont de metodele prezentate şi de altele din literatura de specialitate [A1],[A3],[B1],[H3],[K4],[K7],[S1],[Y1], se pot trage următoarele concluzii :

- utilitatea şi eficacitatea unei metode este direct influenţată de tipul şi complexitatea schemei logice asupra căreia se aplică;
- în general, volumul de calcule solicitat este mare;
- pentru cazul schemelor secvenţiale, problema determinării vectorilor de testare este foarte greu de rezolvat;
- este indicat să se abordeze acele metode care sînt uşor de programat, pentru a putea fi utilizate în cazul construirii automate a vectorilor de testare.

In această lucrare se prezintă o metodă cu aplicabilitate mai largă, destinată testării schemelor logice combinaţionale şi a celor secvenţiale sincrone, în care există, la un moment dat, un singur defect ce nu schimbă tipul schemei (de exemplu, din schemă combinaţională să devină schemă secvenţială asincronă). Se iau în considerare numai defectele permanente, de tipul forţărilor la "0" logic (f0) sau la "1" logic (f1). Această clasă de defecte acoperă majoritatea tipurilor de defecte ce pot afecta comportarea unui circuit logic. In general, în cazul unui astfel de circuit din familia TTL, pot apare următoarele defecte :

- linie de intrare întreruptă sau scurtcircuitată cu sursa de alimentare - este asimilat ca defect fl, la intrare;
- linia de intrare scurtcircuitată la masă - este asimilat ca defect fo, la intrare;
- întreruperea legăturii la sursa de alimentare sau la masă - sînt asimilate ca defecte fl, la ieşire;
- etajul final întrerupt - poate fi asimilat ca defect fl sau fo, la ieşire, în funcţie de locul defectului şi, deci, după comportarea ieşirii;
- scurtcircuite între liniile de intrare (pot fi generate de scurtcircuite între două ieşiri anterioare) - pot fi asimilate ca defecte fl sau fo şi detectate cu aceiaşi vectori de testare ca şi în cazul defectelor clasice fl sau fo, dar aplicaţi într-o anumită succesiune.

Metoda poate fi extinsă şi în unele cazuri de existenţă a defectelor multiple. Prezentarea detaliată se face în cap.2.

Metoda urmăreşte stabilirea unei combinaţii de semnale, care aplicate pe intrările schemei să realizeze următoarele :

- descoperirea defectului la locul apariţiei lui;
- existenţa defectului să fie pusă în evidenţă la cel puţin o ieşire a schemei logice, adică comportarea elementului defect trebuie transmisă, din aproape în aproape, la o ieşire a schemei. Calea pe care se transmite aceasta, numită cale activată, cuprinde alte circuite presupuse corecte, precum şi legăturile dintre ele.

Combinaţia de semnale care răspunde acestor deziderate se numeşte vector de testare.

Totalitatea vectorilor de testare, destinaţi detectării defectelor dintr-o schemă logică, aranjaţi într-o succesiune stabilită în funcţie de configuraţia şi tipul schemei, constituie programul de testare.

Descrierea comportării schemei logice în prezenţa unui defect, precum şi a posibilităţii de influenţare a unei ieşiri de către respectivul defect, se poate realiza cu ajutorul modulului primitiv abstract, MPA, elaborat pentru fiecare tip de circuit, în parte.

MPA se construieşte pe baza relaţiilor dintre intrările, ieşirile şi stările circuitului şi cuprinde tabelul de adevăr, TA şi tabelul de evidenţiere a defectelor, TED.

TED, pentru un circuit dat, conţine combinaţiile de semnale, care trebuie aplicate pe intrările sale, astfel încît orice defect permanent, de tipul fl sau fo, ce apare în circuitul respectiv, să poată fi evidenţiat la cel puţin o ieşire a sa.

Realizarea MPA pentru diferitele tipuri de circuite logice SSI, MSI şi LSI se prezintă în cap.2.

In cap.3 se prezintă un sistem de programe, destinat stabilirii automate, a succesiunii în execuţie, a vectorilor de testare.

Aplicarea vectorilor de testare asupra plachetei ce se verifică şi înregistrarea răspunsurilor acesteia, se face cu ajutorul unui sistem automat de testare. Configuraţia propusă şi realizată, pentru un sistem automat de testare, se prezintă în cap. 4. Se descriu modulele componente ale sistemului, precum şi modul de interconectare şi exploatare ale acestora.

In cap.5 se prezintă concluziile finale ale lucrării.

*
* *

Pentru conducerea şi sprijinul deosebit de competent, acordat permanent pe tot parcursul activităţii de doctorat, autorul mulţumeşte şi pe această cale şi doreşte să-şi exprime stima şi respectul tov.prof.dr.ing. Rogojan Alexandru, conducătorul ştiinţific.

Autorul ţine să-şi exprime întreaga sa recunoştinţă faţă de tov.prof.dr.ing.Hăngănuţ Marius, şeful filialei I.P.A.T.C.T., care l-a ajutat şi îndrumat pe tot parcursul activităţii de doctorat.

Este locul şi momentul ca autorul să declare, cu convingere şi deplină satisfacţie, că şi elaborarea acestei lucrări, ca oricare alta de altfel, evidenţiază condiţia succesului în activitatea ştiinţifică : sprijinul şi colaborarea celor din jur. Autorul le mulţumeşte astfel, prietenilor şi colegilor de la IPATCT filiala Cluj-Napoca şi de la fac.de electrotehnică a Institutului politehnic Cluj-Napoca, cu care a colaborat şi speră să colaboreze şi în continuare, în condiţii din cele mai bune.

Autorul doreşte să mulţumească conducerii fac.de electrotehnică a Inst.politehnic Cluj-Napoca pentru sprijinul acordat în toată activitatea ce o desfăşoară în această instituţie.

Cap.2. METODĂ DE TESTARE A PLACHETELOR ECHIPATE CU CIRCUITE LOGICE SSI, MSI ŞI LSI

2.1. Consideraţii generale

Complexitatea crescîndă a schemelor realizate cu circuite logice influenţează negativ testarea plachetelor pe care se realizează aceste scheme.

Un defect poate afecta comportarea unei intrări, a unei ieşiri sau starea unui circuit logic din componenţa schemei cu care este echipată placheta. Pentru a-l putea descoperi, este necesar ca influenţa lui să se observe la cel puţin o ieşire accesibilă a schemei. Intrările, respectiv ieşirile accesibile sînt cele legate direct de pinii conectoarelor.

Descoperirea defectului şi evidenţierea influenţei lui la o ieşire accesibilă se face prin aplicarea, pe intrările accesibile, a unei combinaţii de semnale potrivite, descrisă prin vectorul de testare. Defectul, pentru care nu se poate construi cel puţin un astfel de vector de testare, este nedetectabil. Este necesar un algoritm ce să construiască vectorii de testare pentru fiecare defect în parte. Acest lucru este posibil dacă există o descriere potrivită a circuitelor logice ce intră în componenţa schemei. Se apreciază că o descriere sub forma MPA satisface această cerinţă.

Metoda de testare ce se propune cuprinde :

- realizarea MPA pentru diferitele tipuri de circuite logice;
- elaborarea algoritmului de construire a vectorilor de testare.

Se introduc următoarele definiţii :

Definiţia 1. Un defect este observabil la o ieşire, dacă există o combinaţie de semnale, care, aplicată pe intrările schemei logice, determină răspunsuri diferite la acea ieşire, în cazul prezenţei, respectiv absenţei defectului.

Definiţia 2. Un defect este controlabil, dacă există o combinaţie de semnale, care, aplicată pe intrările schemei, de-

termină, la locul defectului, o stare logică opusă celei cauzată de defect.

Deci, pentru ca o combinaţie de semnale aplicate pe intrările schemei logice să fie caracterizată ca şi vector de testare, trebuie să i se poată asocia cel puţin un defect controlabil şi observabil.

Dacă prin variabila logică "P" se notează valoarea, la un moment dat, a unei intrări, a unei ieşiri sau a stării unui circuit logic din schemă, observarea acestei valori la o ieşire accesibilă este asigurată, dacă variabila P sau $\bar{P}$ se transmite pînă la respectiva ieşire. Regulile de operare ce caracterizează variabila P sînt :

$$\begin{aligned} &P + P = P \;; && \bar{\bar{P}} = P \;; \\ &P + 1 = 1 \;; && P + 0 = P \;; \\ &P \cdot 1 = P \;; && P \cdot 0 = 0 \;. \end{aligned} \qquad (2.1)$$

Pe tot parcursul unei căi activate este obligatorie îndeplinirea relaţiilor :

$$\begin{aligned} &P + 0 + 0 + \text{---} + 0 = P \;; \\ &P \cdot 1 \cdot 1 \cdot \text{---} \cdot 1 = P \;. \end{aligned} \qquad (2.2)$$

In cazul existenţei unui defect, valoarea variabilei P, transmisă de la locul defect, la o ieşire accesibilă, va fi opusă celei obţinută în absenţa defectului.

Avantajele utilizării acestei notaţii se prezintă cu ajutorul exemplului din fig.2.1.

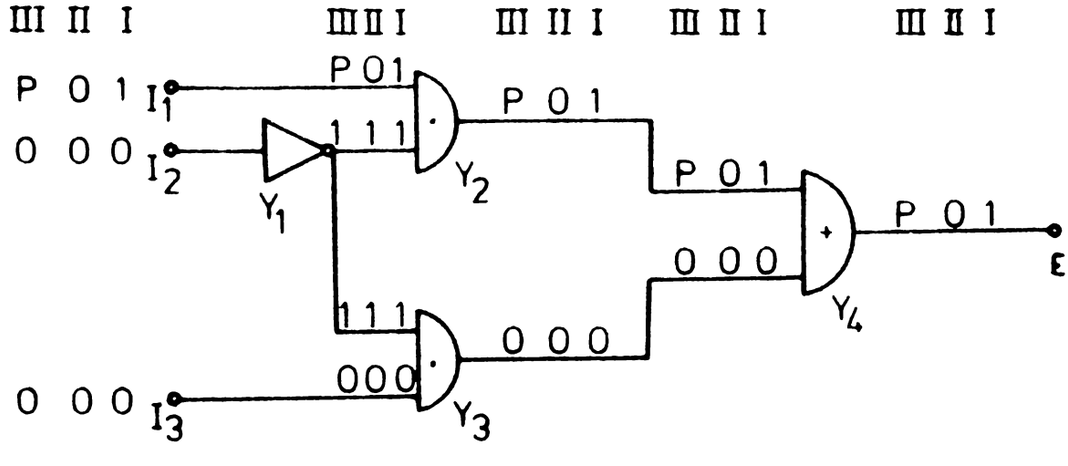


Fig.2.1.

Dacă pe intrările I_1, I_2 şi I_3 se aplică combinaţia de semnale 1oo, valoarea răspunsului obţinut la ieşire este egală cu "1" logic. Altfel spus, se observă că valoarea "1" a semnalului aplicat pe intrarea I_1 se transmite prin circuitele Y_2 şi Y_4 pînă la ieşire (cazul I).

Dacă semnalele aplicate pe intrările I_2 şi I_3 îşi păstrează valorile, iar semnalul aplicat pe intrarea I_1 devine "0" logic, răspunsul obţinut la ieşire va avea valoarea "0" logic. Această valoare "0" este cea aplicată pe intrarea I_1 şi transmisă prin circuitele Y_2 şi Y_4 sau cea care circulă prin circuitele Y_3 şi Y_4 (cazul II).

Notînd cu "P" valoarea logică a semnalului aplicat pe intrarea I_1, la un moment dat, se pune în evidenţă foarte clar, care este intrarea ale cărei valori se transmit la ieşire (intrarea ale cărei valori influenţează direct valoarea ieşirii), precum şi calea pe care se transmit aceste valori (cazul III).

In cazul existenţei unui defect pe calea respectivă, variabila P va primi o valoare opusă celei avute în absenţa defectului. Astfel, analizînd valoarea variabilei P obţinută la ieşire, se poate evidenţia, sigur, existenţa unui defect pe calea parcursă de această variabilă.

Tabelul de evidenţiere a defectelor, TED, realizat pentru fiecare tip de circuit logic conţine, pe fiecare linie, combinaţia de semnale, care aplicată pe intrările circuitului logic, determină transmisia, pînă la o ieşire a circuitului, a variabilei logice P. In felul acesta se poate observa, la o ieşire a circuitului, orice defect ce apare pe o intrare, o ieşire sau într-o stare a sa.

Dacă se activează, la un moment dat, mai multe căi, variabila P se transmite simultan pe toate căile activate. Procesul se numeşte activare multiplă.

Se presupune că schema logică ce se testează nu este redundantă, deci pentru fiecare defect există cel puţin un vector de testare, iar la un moment dat, în schemă, apare un singur defect.

Se iau în considerare numai defectele permanente de tipul forţărilor la "0" logic (fo) şi la "1" logic (f1).

Un defect f_1 sau f_o pe o linie redundantă este nedetectabil. Totodată, dacă există un defect pe liniile redundante şi un alt defect pe celelalte linii, primul poate bloca acţiunea vectori-

lor de testare destinaţi celui de-al doilea defect.

Se vor face unele consideraţii şi asupra defectelor multiple.

Dintre celelalte tipuri de defecte, cele mai frecvente, care apar pe plachetele echipate, sînt scurtcircuitele între pini sau trasee $|*0|$, $|*1|$ şi pot afecta valorile semnalelor aplicate pe intrările unui circuit sau răspunsul obţinut la ieşirea unui circuit. Existenţa unui astfel de defect poate bloca acţiunile unor vectori de testare, destinaţi defectelor de tipul fl şi fo. Totodată, acest tip de defecte poate introduce legături suplimentare, care să transforme tipul schemei logice (de exemplu, din schemă combinaţională devine schemă secvenţial asincronă). Vectorii de testare, care urmăresc detectarea şi a acestor tipuri de defecte (asimilîndu-le cu defectele clasice fl şi fo), sînt mai complexi, iar aplicarea lor asupra intrărilor trebuie să se facă într-o anumită succesiune.

Datorită acestor motive este indicat ca operaţia de testare să debuteze cu verificarea scurtcircuitelor. Pentru aceasta s-au elaborat echipamente de testare specializate.

Definiţia 3. Două defecte sînt echivalente dacă funcţia logică realizată la ieşirea schemei este aceeaşi, indiferent care din cele două defecte este prezent în schemă. De exemplu un defect de tipul fo, la o intrare a unui circuit logic SI-NU, este echivalent cu un defect de tipul fl, de la ieşirea aceluiaşi circuit logic SI-NU. Este foarte importantă determinarea defectelor echivalente, deoarece aceasta are influenţă directă asupra reducerii numărului vectorilor de testare. Asupra acestei probleme se va reveni în cap.3.

Se mai precizează faptul că redundanţa poate transforma două defecte distincte în defecte echivalente.

Orice circuit logic se poatedescriesub forma unui MPA, utilizînd următoarele noţiuni :

- intrări de date, I_D;
- intrări de comandă, I_{cd};
- ieşiri de date, Z_D;
- ieşiri de control, Z_c;
- stările circuitului logic, Y_t^*.

In paragraful următor se prezintă realizarea MPA pentru diferitele tipuri de circuite logice.

2.2. Realizarea MPA

2.2.1. Realizarea MPA pentru circuitele logice SSI

Circuitele logice combinaţionale SSI conţin intrări şi ieşiri de date. Realizarea MPA pentru acest tip de circuite se prezintă pe exemplul din fig.2.2, care conţine tabelul de adevăr, TA, al unui circuit logic SI cu trei intrări (a) şi tabelul său de evidenţiere a defectelor, TED.

Prin "X" se notează un semnal a cărui valoare logică este indiferentă.

TED se realizează pe baza tabelului de adevăr şi conţine, în fiecare linie, combinaţia de semnale ce trebuie aplicată pe intrări, astfel încît variabila P să fie transmisă de la o intrare, la ieşire. La ieşire, aceasta poate apare cu valoare adevărată sau negată (P sau $\bar{P}$)

I_{D1}	I_{D2}	I_{D3}	Z_D
0	X	X	0
X	0	X	0
X	X	0	0
1	1	1	1

a)

I_{D1}	I_{D2}	I_{D3}	Z_D
P	1	1	P
1	P	1	P
1	1	P	P

b)

Fig.2.2.

Modul de construcţie al TED pentru celelalte circuite combinaţionale SSI se face identic cu cel prezentat în fig.2.2.

In continuare se vor aborda circuitele integrate bistabile.

Prin Q_t se notează starea ieşirii la momentul t, iar prin Q_{t+1}, starea aceleiaşi ieşiri la momentul t+1. Pentru a pune în evidenţă faptul că circuitul bistabil este corect, trebuie ca, plecînd de la starea la momentul t, Q_t, să rezulte starea la momentul t+1, Q_{t+1}, cu valorile prezentate în fig.2.3.

Se observă că în toate cazurile, valoarea finală a ieşirii circuitului bistabil (Q_{t+1}) este bine precizată ("1" sau "0").

554556
332 H

Qt	Qt+1
0	0
0	1
1	0
1	1
X	1
X	0

Fig.2.3.

De această observaţie se va ţine seama în construcţia combinaţiilor de semnale ce se aplică pe intrările circuitului, pentru a asigura transmiterea variabilei P la ieşire.

Dacă pentru stabilirea valorii stării finale nu interesează valoarea stării iniţiale, aceasta din urmă se notează cu variabila "X". Dar dacă este nevoie să se pună în evidenţă faptul că circuitul şi-a schimbat starea, trebuie să se impună o valoare anumită pentru starea iniţială. In acest caz, variabilei "X" i se ataşează notaţia "0" sau "1", prin care se precizează valoarea logică necesară pentru starea iniţială. De exemplu "X0" va preciza că starea iniţială nu influenţează valoarea stării finale ("X"), dar pentru evidenţierea efectuării unei comutări ce are ca rezultat schimbarea valorii logice a stării iniţiale, aceasta trebuie să aibă valoarea logică "0". Dacă valoarea impusă stării iniţiale este "1", notaţia utilizată va fi "X1".

In cazul circuitelor secvenţiale, posibilitatea stabilirii unei stări iniţiale cunoscute este un element foarte important în efectuarea operaţiei de testare. TED pune, direct, în evidenţă intrările care trebuie acţionate pentru ca circuitul să treacă într-o stare cunoscută, indiferent de starea anterioară.

2.2.1.1. Circuitul bistabil RS asincron

MPA al acestui tip de circuit se prezintă prin tabelul de adevăr (a) şi tabelul de evidenţiere a defectelor (b) din fig. 2.4.

Din tabelul de adevăr se observă că starea finală a ieşirii este bine precizată numai dacă, pe intrările R şi S, se aplică combinaţiile de semnale logice $(1,0)$ sau $(0,1)$.

Ţinînd cont de această observaţie, TED se va construi astfel încît pe fiecare linie să conţină combinaţia de semnale, care aplicate pe intrări, să asigure transmisia variabilei P sau $\bar{P}$ la

I_D	I_{cd}		Y^*		Z_D	
	R	S	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
	1	1	X	$\bar{X}$	X	$\bar{X}$
	1	0	X	$\bar{X}$	1	0
-	0	1	X	$\bar{X}$	0	1

a)

I_D	I_{cd}		Y^*		Z_D	
	R	S	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
—	1	$\bar{P}$	X0	$\bar{X}0$	P	$\bar{P}$
-	$\bar{P}$	1	X1	$\bar{X}1$	$\bar{P}$	P
-	1	1	P	$\bar{P}$	P	$\bar{P}$
—	1	1	$\bar{P}$	P	$\bar{P}$	P

b)

Fig.2.4.

ieşire. Variabila P sau $\bar{P}$ caracterizează comportarea unei intrări sau starea circuitului bistabil.

Analizînd TED se observă că, dacă pe intrările I_{cd} se aplică combinaţia $(1,\bar{P})$ sau $(\bar{P},1)$, circuitul bistabil trece într-o stare finală a cărei valoare logică este bine precizată. Pentru comutarea dintr-o stare iniţială cu valoare logică cunoscută "X0", respectiv "X1", într-o altă stare finală şi revenirea în starea iniţială, este necesar ca pe intrările I_{cd} să se aplice secvenţa $(1,\bar{P};\bar{P},1)$, respectiv $(\bar{P},1;1,\bar{P})$, în funcţie de starea iniţială. Practic, se poate afirma că intrarea a cărei valoare logică este notată cu $\bar{P}$ controlează direct valoarea logică a ieşirii, deci există o cale activată de la respectiva intrare la ieşire, pe care se transmite variabila $\bar{P}$. Însă realizarea acestui control nu se poate face prin aplicarea unei singure combinaţii de semnale pe intrări, ca şi în cazul circuitelor combinaţionale, ci utilizînd o secvenţă de combinaţii, bine stabilită, cum este cazul secvenţelor anterior prezentate ($(1,\bar{P};\bar{P},1)$ sau $(\bar{P},1;1,\bar{P})$).

Acest lucru rămîne valabil în cazul tuturor circuitelor secvenţiale, însă secvenţa de combinaţii, prin care să se asigure un control eficient, creşte cu complexitatea circuitului şi este influenţată de tipul acestuia.

Dacă variabila P sau $\bar{P}$ caracterizează starea internă a circuitului bistabil, transmisia ei la ieşire se asigură dacă pe intrările I_{cd} se aplică combinaţia $(1,1)$.

Dacă starea iniţială nu este cunoscută (X), secvenţa ce se aplică pe intrări va fi $(1,\bar{P};\bar{P},1;1,\bar{P})$, respectiv $(\bar{P},1;1,\bar{P};\bar{P},1)$,

prima combinaţie din fiecare secvenţă fiind necesară pentru aducerea într-o stare iniţială cu valoare logică cunoscută.

După cum se observă în TED, la o ieşire a circuitului bistabil se transmite variabila $P(\bar{P})$ iar la cealaltă ieşire, variabila $\bar{P}(P)$. Dacă circuitul bistabil se utilizează într-o schemă logică şi nu se doreşte transmiterea în continuare, atît a variabilei $P(\bar{P})$ cît şi a variabilei $\bar{P}(P)$, se poate bloca una dintre căile de transmisie, utilizînd relaţiile (2.1). Această observaţie este valabilă şi în cazul circuitelor secvenţiale MSI şi LSI.

In concluzie, se poate aprecia că TED răspunde, în totalitate, necesităţilor de testare ale circuitului bistabil RS asincron.

2.2.1.2. Circuitul bistabil D

In cazul circuitului bistabil D sincron, se consideră ca şi intrări de comandă, Icd, pe lîngă intrarea de tact, TC, şi intrările asincrone R şi S, prioritare faţă de cele sincrone. S-a adoptat acest mod de lucru, deoarece este caracteristic circuitelor bistabile integrate şi apare în cazul majorităţii circuitelor integrate secvenţiale MSI şi LSI.

Tabelul de adevăr se prezintă în fig.2.5.(a), iar tabelul de evidenţiere a defectelor, în fig.2.5.(b). Pentru a nu încărca TED, modul de lucru aferent intrărilor asincrone R şi S nu se va mai relua, acesta fiind identic cu cel din cazul circuitului bistabil RS asincron. Insă în faza de testare, pentru o descriere completă a unui circuit bistabil D sincron, care conţine şi intrări asincrone, TED trebuie completat cu liniile aferente acestor tipuri de intrări.

Pe intrarea de tact, TC, se pot aplica semnale de tip nivel logic, cu valorile "1", "0", "X", sau semnale de tact, notate cu T.

Din analiza TED se observă că variabila logică P sau $\bar{P}$ se transmite la ieşire, de la fiecare intrare (D şi TC). Dacă această variabilă caracterizează starea internă a bistabilului, transmisia ei la ieşire este asigurată, dacă pe intrarea de tact, TC, se aplică un semnal cu valoarea logică "0". Deci pentru fiecare linie din TED există o cale activată pînă la ieşire.

Comutarea dintr-o stare iniţială cunoscută, 'X0' sau 'X1', într-o altă stare finală, se obţine acţionînd pe intrările D şi TC.

I_D	I_{cd}			Y^*		Z_D	
D	TC	R	S	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
X	X	0	1	X	$\bar{X}$	0	1
X	X	1	0	X	$\bar{X}$	1	0
1	T	1	1	X	$\bar{X}$	1	0
0	T	1	1	X	$\bar{X}$	0	1
X	0	1	1	X	$\bar{X}$	X	$\bar{X}$

a)

I_D	I_{cd}			Y^*		Z_D	
D	TC	R	S	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
1	P	1	1	X0	$\bar{X}0$	P	$\bar{P}$
P	T	1	1	X0	$\bar{X}0$	P	$\bar{P}$
$\bar{P}$	T	1	1	X1	$\bar{X}1$	$\bar{P}$	P
X	0	1	1	P	$\bar{P}$	P	$\bar{P}$
X	0	1	1	$\bar{P}$	P	$\bar{P}$	P

b)

Fig.2.5.

2.2.1.3. Circuitul bistabil JK

Intrările de comandă, I_{cd}, sînt, ca şi în cazul anterior, TC, R şi S, iar intrările de date sînt J şi K.

Tabelul de adevăr se prezintă în fig.2.6.(a), iar tabelul de evidenţiere a defectelor, în fig.2.6.(b). Acesta din urmă conţine linii aferente numai modului de lucru sincron, cele pentru modul de lucru asincron sînt identice cu liniile construite în cazul bistabilului RS asincron.

I_D		I_{cd}			Y^*		Z_D	
K	J	TC	R	S	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
X	X	0	0	1	X	$\bar{X}$	0	1
X	X	X	1	0	X	$\bar{X}$	1	0
0	1	T	1	1	X	$\bar{X}$	1	0
1	0	T	1	1	X	$\bar{X}$	0	1
0	0	T	1	1	X	$\bar{X}$	Q_t	$\bar{Q}_t$
1	1	T	1	1	X	$\bar{X}$	$\bar{Q}_t$	Q_t
X	X	0	1	1	X	$\bar{X}$	X	$\bar{X}$

a)

I_D		I_{cd}			Y^*		Z_D	
K	J	TC	R	S	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
0	1	P	1	1	X0	$\bar{X}0$	P	$\bar{P}$
1	0	P	1	1	X1	$\bar{X}1$	$\bar{P}$	P
0	P	T	1	1	X0	$\bar{X}0$	P	$\bar{P}$
P	0	T	1	1	X1	$\bar{X}1$	$\bar{P}$	P
1	1	T	1	1	P	$\bar{P}$	$\bar{P}$	P
1	1	T	1	1	$\bar{P}$	P	P	$\bar{P}$
X	X	0	1	1	P	$\bar{P}$	P	$\bar{P}$
X	X	0	1	1	$\bar{P}$	P	$\bar{P}$	P

b)

Fig.2.6.

Si în acest caz, fiecare linie din TED realizează o cale activată pe care să fie transmisă la ieşire, variabila P sau $\bar{P}$. Această variabilă caracterizează comportarea unei intrări (K, J sau TC) sau starea circuitului bistabil (Y).

Totodată, se pun în evidenţă şi combinaţiile care trebuie aplicate pe intrări, pentru a aduce circuitul într-o stare cunoscută, indiferent de starea sa anterioară.

2.2.2. Realizarea MPA pentru circuitele logice MSI

2.2.2.1. Realizarea MPA pentru circuitele logice MSI combinaţionale.

Modul de lucru este asemănător cu cel din cazurile precedente. Deoarece circuitele logice MSI sînt mai complexe şi TED aferente vor fi mai ample.

Din clasa circuitelor integrate combinaţionale se prezintă realizarea MPA pentru cazul circuitului de multiplexare SN74150. Deoarece întocmirea tabelului de adevăr nu ridică probleme deosebite, se renunţă la prezentarea lui. In fig.2.7 se prezintă tabelul de evidenţiere a defectelor, pentru acest tip de circuit.

Intrările de date, I_D, sînt notate D_0-D_{15}, iar intrările de comandă sînt : intrările de selecţie S_0-S_3 şi intrarea de validare E.

Din TED se observă că variabila P sau $\bar{P}$ este transmisă la ieşire de la oricare din intrările circuitului. Astfel, aplicînd o combinaţie de semnale potrivite pe celelalte intrări, se asigură ca intrarea pe care este aplicat semnalul, notat cu variabila P, să controleze, la momentul respectiv, valoarea ieşirii, indiferent de valoarea logică a variabilei P($\bar{P}$).

2.2.2.2. Realizarea MPA pentru circuitele logice MSI secvenţiale

Realizarea MPA în cazul registrelor se prezintă pentru registrul de 4 biţi, CDB495.

Intrările de date, I_D, sînt : I_{DS} - intrarea serială, $I_{DP_{1-4}}$ - intrări paralele. Intrările de comandă, I_{cd}, sînt : ML - modul de lucru al registrului, TP - intrarea de tact pentru încărcare paralelă/deplasare stînga, TS - intrarea de tact pentru încărcare serie/deplasare dreapta.

ID									Icd					ZD
D_0	D_1	D_2		D_4		D_7		D_{15}	S_3	S_2	S_1	S_0	E	--
X	X	X		X		X		X	X	X	X	X	P	P
0	X	X		X		X		X	0	0	0	$\bar{P}$	0	P
1	X	X		X		X		X	0	0	0	$\bar{P}$	0	$\bar{P}$
X	0	X		X		X		X	0	0	0	P	0	P
X	1	X		X		X		X	0	0	0	P	0	$\bar{P}$
X	X	0		X		X		X	0	0	P	0	0	P
X	X	1		X		X		X	0	0	P	0	0	$\bar{P}$
X	X	X		0		X		X	0	P	0	0	0	P
X	X	X		1		X		X	0	P	0	0	0	$\bar{P}$
X	X	X		X		0		X	P	0	0	0	0	P
X	X	X		X		1		X	P	0	0	0	0	$\bar{P}$
P	X	X		X		X		X	0	0	0	0	0	$\bar{P}$
$\bar{P}$	X	X		X		X		X	0	0	0	0	0	P
X	P	X		X		X		X	0	0	0	1	0	$\bar{P}$
X	$\bar{P}$	X		X		X		X	0	0	0	1	0	P
X	X	P		X		X		X	0	0	1	0	0	$\bar{P}$
X	X	$\bar{P}$		X		X		X	0	0	1	0	0	P
X	X	X		P		X		X	0	1	0	0	0	$\bar{P}$
X	X	X		$\bar{P}$		X		X	0	1	0	0	0	P
X	X	X		X		P		X	1	0	0	0	0	$\bar{P}$
X	X	X		X		$\bar{P}$		X	1	0	0	0	0	P
X	X	X		X		X		P	1	1	1	1	0	$\bar{P}$
X	X	X		X		X		$\bar{P}$	1	1	1	1	0	P

Fig.2.7.

Tabelul de adevăr se prezintă în fig.2.8, iar tabelul de evidenţiere a defectelor, în fig.2.9.

Numărul şi tipul operaţiilor ce se realizează sînt proprii registrului ce se analizează. Pentru fiecare din operaţiile ce caracterizează funcţionarea registrului se urmăreşte aplicarea pe intrările de date, respectiv de comandă, a unei astfel de combinaţii de semnale, care să asigure transmisia variabilei P sau $\bar{P}$ la cel

puţin o ieşire.

I_D		I_{cd}			$Y_i^*, i=1,4$	$Z_{Dj}, i=1,4$	Operatia realizată
I_{DS}	$I_{DPi}, i=1,4$	ML	TP	TS	-	-	-
X	I_{DPi}	1	T	X	X	I_{DPi}	Încărcare paralelă, IP
I_{DS}	X	0	X	T	Y_i	$Z_{Di} \leftarrow Y_{i-1}$ $Z_{D1} \leftarrow I_{DS}$	Încărcare serie/deplasare dreapta, IS/DD
X	X	1	0	X	Y_i	Y_i	Stare neschimbată, SN
X	X	0	X	0	Y_i	Y_i	Stare neschimbată, SN
X	X	X	0	0	Y_i	Y_i	Stare neschimbată, SN

Fig.2.8.

I_D		I_{cd}			Y_{i-1}^*	Y_i^*	$Z_{D_{i-1}}$	Z_{Di}	Operatia realizata
I_{DS}	I_{DPi}	M L	T P	T S	-	-	-	-	-
X	1	1	P	X		X0		P	IP
X	0	1	P	X		X1		$\bar{P}$	IP
X	1	P	T	0		X0		P	IP
X	0	P	T	0		X1		$\bar{P}$	IP
X	P	1	T	X		X0		P	IP
X	$\bar{P}$	1	T	X		X1		$\bar{P}$	IP
1	X	0	X	P	X 0		P		IS/DD
0	X	0	X	P	X 1		$\bar{P}$		IS/DD
X	X	0	X	P	0	X1		$\bar{P}$	IS/DD
X	X	0	X	P	1	X0		P	IS/DD
X	X	1	0	X		P		P	SN
X	X	0	X	0		P		P	SN
X	X	X	0	0		P		P	SN

Fig.2.9.

Plecînd de la un set de operaţii stabilite cu ajutorul intrărilor I_D şi I_{cd}, se poate construi TED pentru orice alt tip de registru.

In general, modul de lucru al unui registru dat se descrie cu ajutorul operaţiilor, astfel :

- $(IP)\cdot(DD)^2$ - determină o încărcare paralelă, urmată de două deplasări spre dreapta;
- $(IS)\cdot(DD)^3$ - determină o încărcare serie, urmată de trei deplasări spre dreapta. Aceasta poate fi descrierea unui algoritm de inițializare a unui registru de 4 biți, având ca și efect, încărcarea serie a valorii logice "0" și deplasarea ei spre dreapta, prin toate celulele registrului.

Deci, pe baza unui TED ce conține toate operațiile ce se pot efectua într-un registru, se poate construi TED pentru orice tip particular de registru, apelînd, în plus, la un program care să descrie tipurile și succesiunea operațiilor, ce caracterizează funcționarea respectivului registru. In acest caz, deoarece nu mai este necesară memorarea TED pentru toate tipurile de registre, se obține o importantă economie de memorie.

Realizarea TED în cazul numărătoarelor se prezintă pentru cazul circuitului CDB4192. Acesta este un numărător zecimal sincron, care comută pe frontul anterior al semnalului de tact, în repaus, intrarea de tact având valoarea logică "1". Ca intrări de date, I_D, se consideră intrările pentru încărcare paralelă. Intrările de comandă, I_{cd}, sînt : intrarea de inițializare, R, intrarea de încărcare, L, intrarea de numărare în sens crescător, CU și intrarea de numărare în sens descrescător, CD. Ieșirile de control sînt : ieșirea de transport, C și ieșirea de împrumut, B. Ieșirile de date, Z_{Di}, sînt ieșirile paralele ale numărătorului.

Tabelul de adevăr al acestui tip de circuit se prezintă în fig.2.10, iar tabelul de evidențiere a defectelor, în fig.2.11.

Semnificația notațiilor utilizate este următoarea :

- IN - inițializare;
- IP - încărcare paralelă;
- NC - numărare în sens crescător;
- ND - numărare în sens descrescător.

Unele operații ce se efectuează în numărătoare sînt mai complexe decît în cazul registrelor. Este vorba, în special, de operațiile de numărare. Execuția unei operații de numărare în sens crescător se prezintă în schema logică din fig.2.12, unde $b_i, i = 1,n$ reprezintă configurația binară a numărului.

Un exemplu de realizare a unei numărări, pe baza schemei din fig.2.12 :

i = 1101OX1XX11

i+1 = 1101XXXXX00

Numărarea în sens descrescător se poate face după acelaşi principiu.

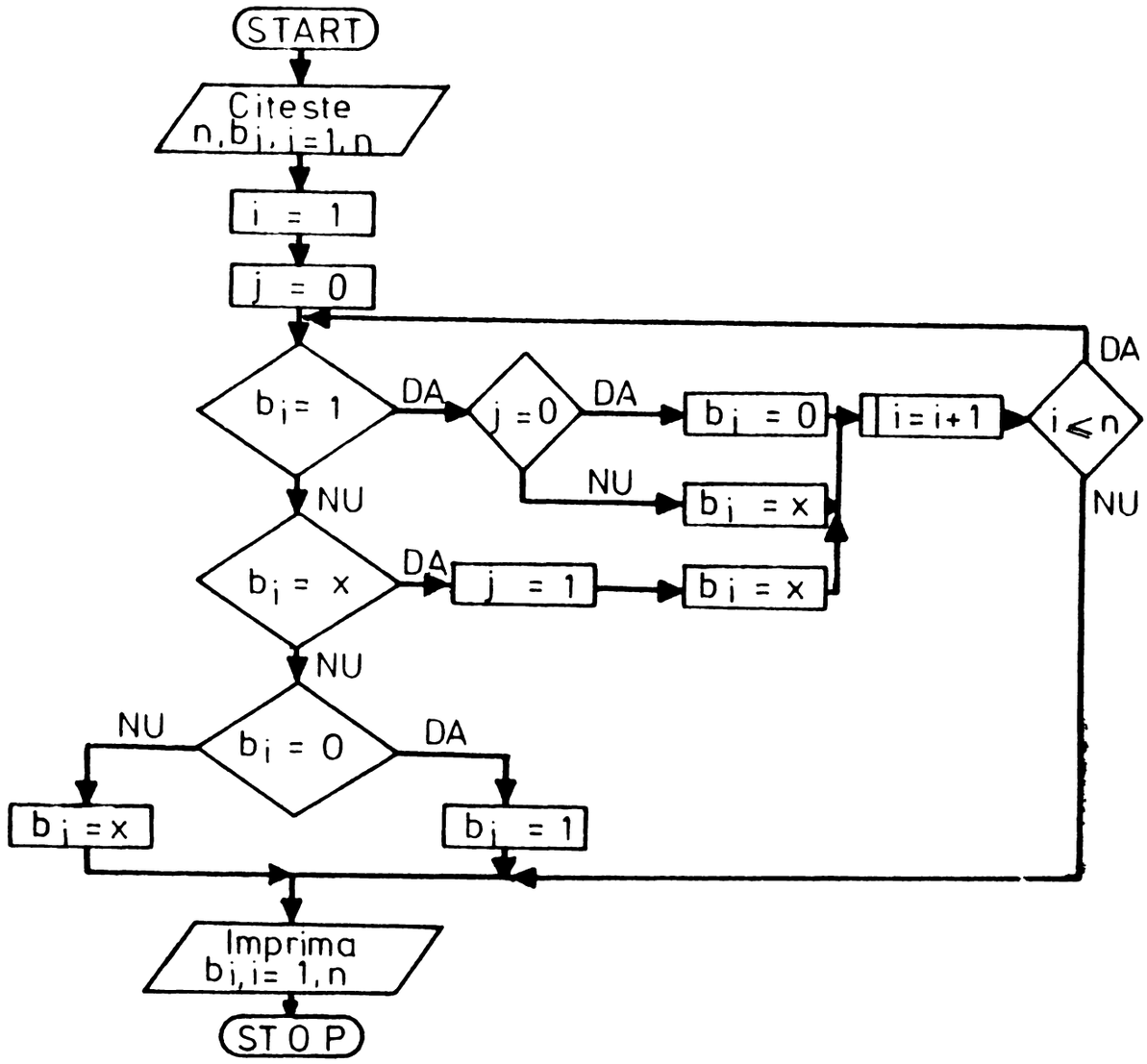


Fig.2.1o.

Ca şi în cazul registrelor, şi aici, plecînd de la un TED ce conţine toate operaţiile care se pot efectua într-un numărător, se poate construi TED pentru orice alt tip de numărător. Pentru fiecare caz în parte, pe lîngă TED existent, se ataşează un program simplu care să stabilească tipul şi succesiunea în execuţie a tuturor operaţiilor ce caracterizează funcţionarea numărătorului dat. In cazul fiecărei operaţii, ce se efectuează în numărător, se va urmări asigurarea transmisiei variabilei P($\overline{P}$)

I_D				I_{cd}				Y_i^*				Z_C		Z_{Di}				Operatia realizata
I_{D1}	I_{D2}	I_{D3}	I_{D4}	R	L	CU	CD	Y_1	Y_2	Y_3	Y_4	B	C	Z_{D1}	Z_{D2}	Z_{D3}	Z_{D4}	-
X	X	X	X	1	X	X	0	X	X	X	X	0	1	0	0	0	0	IN
X	X	X	X	1	X	X	1	X	X	X	X	1	1	0	0	0	0	IN
0	0	0	0	0	0	X	0	X	X	X	X	0	1	0	0	0	0	IN
0	0	0	0	0	0	X	1	X	X	X	X	1	1	0	0	0	0	IN
1	X	X	1	0	0	X	X	X	X	X	X	1	1	1	0	0	0	IP
1	X	X	1	0	0	X	X	X	X	X	X	1	1	1	X	X	1	IP
X	X	X	X	0	1	T	1	0	0	0	0	1	1	1	0	0	0	NC
X	X	X	X	0	1	T	1	1	0	0	0	1	1	0	1	0	0	NC
										⋮								
X	X	X	X	0	1	T	1	1	1	1	0	1	1	0	0	0	1	NC
X	X	X	X	0	1	T	1	0	0	0	1	1	1	1	0	0	1	NC
X	X	X	X	0	1	T	1	1	0	0	1	1	0	0	0	0	0	NC
X	X	X	X	0	1	T	T	1	0	0	1	1	1	0	0	0	1	ND
										⋮								
X	X	X	X	0	1	1	T	0	1	0	0	1	1	1	0	0	0	ND
X	X	X	X	0	1	1	T	1	0	0	0	1	1	0	0	0	0	ND
X	X	X	X	0	1	1	T	0	0	0	0	0	1	1	0	0	1	ND

Fig.2.11.

la cel puţin o ieşire. Această variabilă caracterizează comportarea unei intrări sau starea numărătorului.

In concluzie, pentru orice tip de circuit secvenţial MSI, plecînd de la o mulţime A a operaţiilor, definită cu ajutorul unei mulţimi I a intrărilor de comandă, a unei mulţimi Y^* a stărilor şi a unei mulţimi E a ieşirilor de control, pentru o lungime dată ℓ, se poate elabora TED pentru orice tip de circuit, care să conţină mulţimea operaţiilor $A_1 \in A$, definită pe o mulţime a intrărilor $I_1 \in I$, a stărilor $Y_1^* \in Y^*$ şi a ieşirilor $E_1 \in E$.

I_D	I_{cd}				Y_i^*	$\prod Y_j^*$	$\prod \bar{Y}_j^*$	Z_C		Z_{Di}	Operatia realizata
-	R	L	CU	CD	-	$j<i$	$j<i$	B	C	-	-
X	P	X	X	X	1	X	X	X	X	$\bar{P}$	IN
X	1	X	X	$\bar{P}$	X	X	X	$\bar{P}$	1	0	IN
X	1	X	X	P	X	X	X	P	1	0	IN
1	0	$\bar{P}$	X	X	X	X	X	X	X	P	IP
0	0	$\bar{P}$	X	X	X	X	X	X	X	$\bar{P}$	IP
0000	0	0	X	$\bar{P}$	X	X	X	$\bar{P}$	1	0000	IP
0000	0	0	X	P	X	X	X	P	1	0000	IP
1XX1	0	0	$\bar{P}$	X	X	X	X	1	$\bar{P}$	1XX1	IP
1XX1	0	0	P	X	X	X	X	1	P	1XX1	IP
P	0	0	X	X	X	X	X	X	X	P	IP
$\bar{P}$	0	0	X	X	X	X	X	X	X	$\bar{P}$	IP
X	0	1	T	1	P	0	X	X	X	P	NC
X	0	1	T	1	P	1	X	X	X	$\bar{P}$	NC
X	0	1	T	1	P	X	0	X	X	P	ND
X	0	1	T	1	P	X	1	X	X	$\bar{P}$	ND

Fig.2.12.

2.2.3. Realizarea MPA pentru circuitele logice LSI

Circuitele integrate LSI sînt componente electronice cu dimensiuni de gabarit foarte reduse, în raport cu gradul mare de integrare şi prin aceasta, cu un număr limitat de puncte de acces. Aceste trăsături permit concluzionarea că,anumite funcţii sînt controlate intern, deci odată cu creşterea complexităţii, circuitele LSI devin tot mai ermetice, reducînd accesul sistemului de testare la logica lor internă [L1].

Procesele tehnologice legate de elaborarea circuitelor LSI, în special a microprocesoarelor, nu sînt în totalitate automatizate (de exemplu, generarea măştilor este automatizată numai în ultimele faze) [K6],[H9] . Multitudinea şi complexitatea operaţiilor tehnologice care au loc, coroborate cu imposibilitatea unei supravegheri şi conduceri automate, face ca numărul de defecte, în cazul circuitelor integrate LSI, să fie relativ mare, depăşindu-l cu mult pe cel caracteristic circuitelor integrate MSI sau SSI.

In circuitele LSI se integrează un număr tot mai mare de module, care solicită puţine conexiuni externe, dar prezintă o complexă reţea de conexiuni interne. Se urmăreşte ca raportul dintre numărul de componente, integrate pe acelaşi suport şi numărul de terminale, să fie cît mai mare. Numărul de terminale este limitat, în principal, de dimensiunea suportului, care se urmăreşte să fie cît mai mică. Avantajele acestui mod de realizare sînt : creşterea vitezei de lucru, în urma micşorării lungimii medii a liniilor de conexiuni şi a sarcinilor capacitive, creşterea imunităţii la perturbaţii şi scăderea puterii consumate, în urma dispariţiei amplificatoarelor pe post de emiţătoare de linii [L1]. Dar are ca şi dezavantaj principal, realizarea dificilă a operaţiei de testare şi evaluare a performanţelor, în condiţii reale de funcţionare.

In general, circuitele LSI pot prezenta atît tipurile de defecte "clasice", prezente în cazul circuitelor MSI şi SSI (forţare la "1", forţare la "0", scurtcircuite între linii, intrare în aer etc.), precum şi o categorie de defecte caracteristice regimului de funcţionare dinamic.

In concluzie, se apreciază că circuitele integrate LSI şi plachetele echipate cu astfel de circuite, datorită complexităţii lor deosebite, solicită sisteme şi metode de testare, care să ţină cont de totalitatea particularităţilor ce le caracterizează.

Se utilizează o gamă largă de metode, unele cu aplicabilitate generală pentru orice tip de circuite logice şi plachete echipate cu astfel de circuite, altele proprii acestui tip. Cele mai cunoscute sînt :

- compararea răspunsului cu cel obţinut de la un obiect martor;
- metodele convenţionale (deterministe);
- emularea;
- metode neconvenţionale;
- simularea;
- divizarea în module senzoriale.

a) Utilizarea unui obiect martor, în vederea stabilirii corectitudinii în funcţionare este una dintre cele mai vechi metode de testare. Obiectul martor şi cel care se testează sînt atacaţi cu aceeaşi secvenţă de semnale. Răspunsurile acestora se compară, iar neconcordanţa între ele este un indiciu al existenţei unui defect. Localizarea defectelor se face, în cel mai bun

caz, la nivel de pin al conectorului.

Impedimentul major, ce apare în cazul acestei metode de testare, este necesitatea existenţei unui martor, cu certitudine corect, element greu de realizat şi în ultimă instanţă, neeconomic. Problema se complică dacă plachetele cu circuite LSI prezintă o varietate mare. Metoda este indicată numai în cazul testării în flux tehnologic, cînd numărul de plachete sau circuite de acelaşi tip, care se analizează, este foarte mare [P23].

b) Metodele convenţionale (algoritmul D, metoda Poage, metoda Armstrong etc.) [N9],[*4],[*5] pot fi utilizate pentru testarea plachetelor echipate cu circuite LSI, MSI şi SSI (în marea majoritate a cazurilor, plachetele echipate cu circuite LSI conţin un număr de circuite logice MSI şi SSI), dacă se realizează o descriere potrivită, pentru fiecare tip de circuit. Tehnicile utilizate pînă în prezent, nu permit, în toate cazurile şi într-un mod optim, acest lucru.

Realizarea unei descrieri sub forma MPA, după metodologia prezentată în cazul circuitelor SSI şi MSI poate fi utilizată în cazul oricărui tip de circuit LSI, permiţînd aplicarea metodelor convenţionale pentru testarea plachetelor echipate cu aceste circuite.

Aceste metode nu solicită un obiect martor şi se pretează foarte bine pentru automatizarea procesului de întocmire a programelor de testare.

c) Emularea [H9],[R4],[*4],[*5],[*45] oferă posibilitatea dezvoltării de sisteme realizate cu ajutorul circuitelor LSI, testînd şi evaluînd configuraţia electronică. Totodată permite dezvoltarea sistemului de programe şi verificarea interacţiunii hard-soft.

Un sistem de emulare, prezentat în fig.2.13, cuprinde următoarele elemente : modulul de emulare, ME, modulul condiţiilor de oprire, MCO, modulul de memorare al evenimentelor, MME şi modulul de interfaţă cu utilizatorul, MIU.

Modulul de emulare serveşte pentru :

- înlocuirea microprocesorului din sistemul utilizatorului cu cel existent în ME;
- utilizarea memoriei sistemului de emulare sau a unei părţi din ea, în locul memoriei din sistemul utilizatorului;
- utilizarea modulului MIU, în locul porturilor de intra-

re/ieşire din sistemul utilizatorului. Acest modul se modifică în funcţie de tipul microprocesorului din sistemul utilizator.

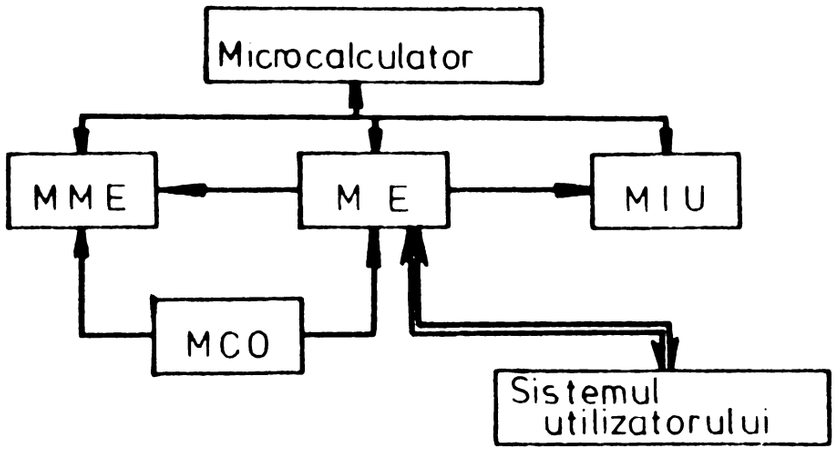


Fig.2.13.

Modulul de memorare al evenimentelor poate înregistra un număr de informaţii aferente diferitelor evenimente, ce apar pe parcursul desfăşurării unui număr de cicluri maşină. In acest fel se oferă o istorie a evoluţiei sistemului pe o anumită perioadă de timp.

Modulul condiţiilor de oprire permite oprirea execuţiei în funcţie de anumite condiţii îndeplinite.

Sistemul de emulare va asigura execuţia programului utilizator, colectînd informaţii despre fiecare instrucţiune. Utilizatorul poate asigna o parte din memoria sistemului de emulare, precum şi resursele sale de intrare/ieşire, elemente care, din punct de vedere logic, devin resurse ale utilizatorului.

Analizînd avantajele şi dezavantajele utilizării unui sistem de emulare în testare, se pot trage următoarele concluzii :

- emularea asigură o testare completă a unui ansamblu realizat pe bază de circuite LSI, din punct de vedere hard,soft şi al interacţiunii acestora;
- un sistem de emulare este întotdeauna orientat spre un anumit tip de microprocesor, deci utilizarea lui în testare este indicată numai pentru serii mari;
- realizarea unui sistem de emulare reprezintă o investiţie importantă, motiv pentru care decizia de realizare se va lua în urma unei analize minuţioase privind dota-

rea, gradul de utilizare, evoluţia tipurilor de microprocesoare ce se vor utiliza etc.;
- un sistem de emulare poate fi utilizat şi pentru dezvoltarea de programe orientate pe tipul microprocesorului cu care este dotat;
- emularea este o tehnică de testare, proprie numai ansamblelor realizate cu circuite LSI programabile.

d) Metodele neconvenţionale [*3], [*1], [J1], [*5], [*21] urmăresc aplicarea asupra elementului care se verifică, a vectorilor de testare ce au caracter pseudo-aleator şi înregistrarea răspunsului comprimat, după o anumită lege, sub forma unei "semnături". In felul acesta, un răspuns cu o lungime, în biţi, oricît de mare, se reduce la 16-2o biţi, aceasta fiind dimensiunea unei "semnături". Deci cantitatea de informaţii ce trebuie înregistrată şi analizată este foarte redusă. Din acest motiv, metoda se pretează la testarea de mare viteză.

Ca şi deficienţă se semnalează faptul că, pentru testare, necesită cunoaşterea semnăturilor de pe placheta corectă, ceea ce, în unele cazuri este dificil de obţinut. Dacă există o plachetă cu certitudine corectă, poate fi utilizată pentru obţinerea semnăturilor corecte.

e) Simularea [N2], [P4] [*4] [*5] este o metodă mult utilizată pentru realizarea programelor de testare şi totodată, ca un instrument pentru facilitarea operaţiei de localizare a defectelor. Unele defecte, cum sînt hazardul, apariţia de impulsuri parazite, pot fi puse în evidenţă numai cu această metodă.

Problema care trebuie rezolvată, pentru a putea utiliza optim această metodă, este ca descrierea schemei logice de pe placheta echipată şi simularea acesteia să se realizeze cît mai simplu şi să nu solicite o capacitate de memorare prea mare. Aceasta se obţine dacă circuitele logice ce formează schema realizată pe o plachetă se descriu cu ajutorul MPA, sub forma prezentată în paragrafele anterioare. Modul de lucru se va prezenta amănunţit în cap.3.

Simularea, în această tehnică, reţine numai aspectul funcţional, nu şi cel electronic. Dar acest element este mai puţin interesant în cazul testării plachetelor echipate.

f) Divizarea în module senzoriale [T2], [V3], [J1], [O2]. Un modul senzorial este o unitate logică complexă, care poate fi ac-

tivată şi testată individual sau utilizînd numai modulele verificate în paşii precedenţi. In felul acesta se urmăreşte ca fiecare modul să fie capabil a-şi tranamite răspunsul său (starea sa), direct sau indirect, pe magistralele externe ale circuitului LSI, cînd pe intrările sale se aplică o combinaţie de semnale.

Activarea constă în stimularea modulului pentru a-l face accesibil, după care se aplică asupra lui o combinaţie de semnale, care determină comutarea dintr-o stare iniţială, într-o stare finală, ce se poate analiza.

Pentru circuitele logice LSI, mai ales în cazul microprocesoarelor, vectorul de testare se construieşte cu ajutorul instrucţiunilor proprii. Cu aceste instrucţiuni se activează fiecare modul ce formează circuitul LSI, astfel încît orice defect ce afectează comportarea modulului să fie descoperit, iar influenţa lui să poată fi observată pe una din magistralele externe ale circuitului LSI. Defectele pot apare pe intrările, ieşirile sau în stările diferitelor module.

Astfel se asigură, implicit, şi o testare a diferitelor module ale circuitului LSI.

Utilizarea acestei metode de testare este obligatorie dacă placheta se verifică cu ajutorul tehnicii de testare "IN CIRCUIT".

Circuitul integrat LSI, fiind numai parţial accesibil, nu toate modulele vor putea fi testate în întregime, existînd cel puţin un modul iniţial, care se consideră corect, eventual a fost testat printr-o altă metodă.

Problemele care apar în cazul utilizării acestei metode sînt următoarele :

- ordonarea instrucţiunilor astfel încît execuţia lor să stabilească modulele senzoriale şi să permită testarea acestora;
- testarea propriu-zisă a modulelor.

Prima problemă solicită o perfectă cunoaştere a întregului set de instrucţiuni sau comenzi/moduri de lucru, pentru circuitul LSI care se testează, ceea ce este greu de realizat, datorită multitudinii şi complexităţii diferitelor tipuri de circuite LSI.

Pentru rezolvarea celei de-a doua probleme este absolut necesară o descriere potrivită a diferitelor module, deoarece, în caz contrar, datorită complexităţii circuitelor LSI, se solicită o mare capacitate de memorare şi o investiţie importantă de

timp,pentru întocmirea programelor de testare. Dacă nu există o astfel de descriere, unele circuite LSI nu se pot testa sau se testează numai parţial.

Dacă se rezolvă optim aceste două probleme, metoda de testare, care divizează circuitul LSI în module senzoriale, este cea mai indicată,atît pentru testarea circuitelor LSI,cît şi a plachetelor echipate cu aceste circuite.

In continuare, pentru rezolvarea acestor probleme, se propune următorul mod de lucru :

- ordonarea automată a setului de instrucţiuni (comenzi etc.) pentru stabilirea şi activarea modulelor senzoriale de tip registru,descrise cu ajutorul MPA, plecînd de la tabelul instrucţiunilor;
- realizarea programelor pentru testarea modulelor senzoriale utilizînd un algoritm convenţional.

Procedeul propus se prezintă în cazul microprocesoarelor, acestea fiind cele mai complexe circuite logice LSI. Exemplificările se vor face pe microprocesorul Intel 8080.

Analizînd o gamă de microprocesoare (I8080, M6800, TMS9900), în fig.2.14, se prezintă o configuraţie generală la nivel de modul de tip registru. Toate modulele componente vor fi descrise ca MPA de tip registru.

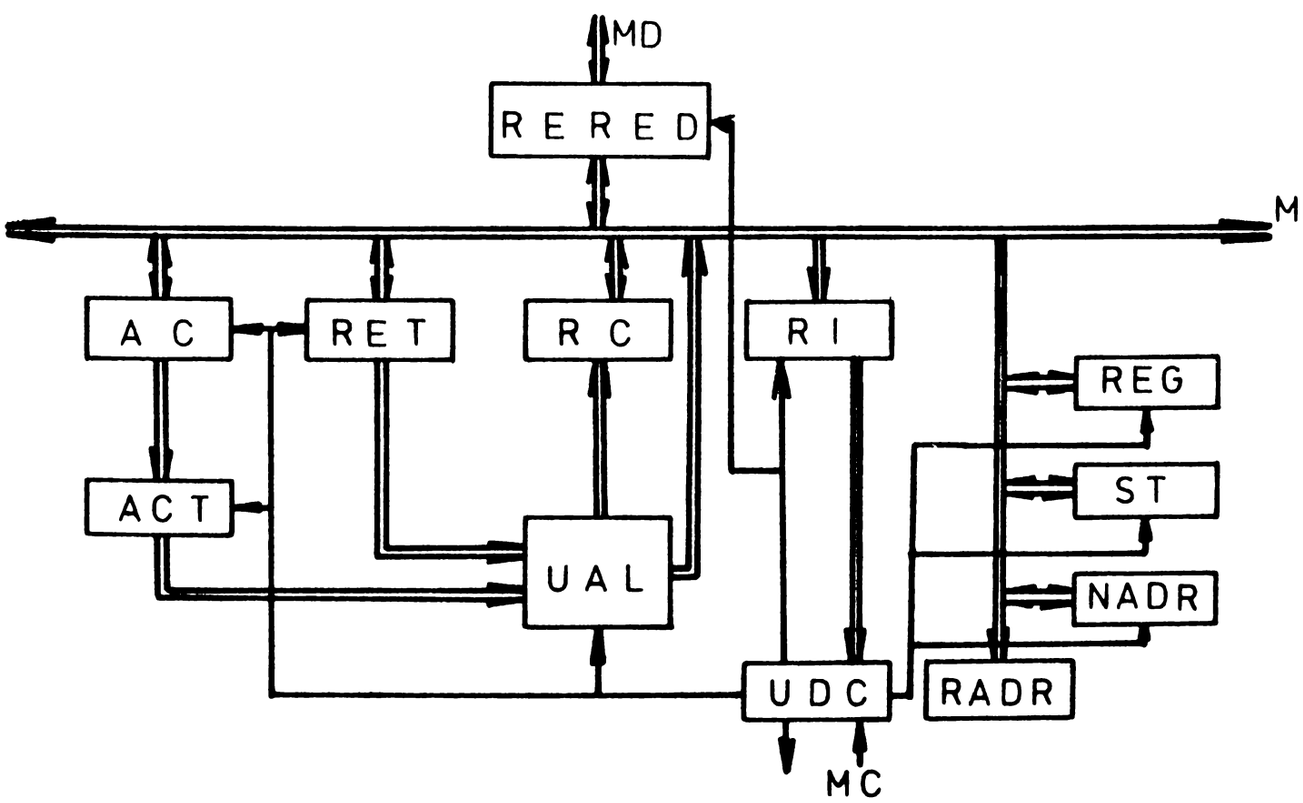


Fig.2.14.

Semnificaţia notaţiilor utilizate este următoarea :

- unitatea de decodificare şi control, UDC;
- registrul de instrucţiuni, RI;
- unitatea aritmetică şi logică, UAL;
- acumulator, AC;
- acumulator temporar, ACT;
- registrul pentru păstrarea temporară a informaţiilor, RET;
- registre generale, REG;
- registrul indicator de stivă, ST;
- numărător de adrese, NADR;
- registru pentru recepţia/emisia informaţiilor (instrucţiuni, date), RERED;
- magistralele de date, adrese, comenzi, internă, MD, MA, MC, MI.

Cu linie simplă se specifică circulaţia semnalelor de comandă.

In faza de realizare a MPA se ţine cont de faptul că registrele au lungimi diferite, iar unele pot fi adresate ca registre perechi. Ţinînd cont de această observaţie, MPA de tip registru ale microprocesorului I8080 sînt :

a) - module cu lungimea de 8 biţi : RERED, RADRH, RADRL, NADRH, NADRL, STH, STL, B, C, D, E, H, L, W, Z, AC, ACT, RET, UAL, RI, Z, CY, S, P, C;

b) - module cu lungimea de 16 biţi :

RADR = (RADRH, RADRL);
NADR = (NADRH, NADRL);
ST = (STH, STL);
BC = (B, C);
DE = (D, E);
HL = (H, L);
WZ = (W, Z).

c) - module suplimentare (fictive), definite cu scopul de a înlesni efectuarea unor operaţii :

- B0 - registru ce conţine, tot timpul, valoarea zero;
- B1 - registru ce conţine, tot timpul, valoarea "1";
- BD - registru de manevră;
- BM - registru încărcat cu "1" în toate poziţiile.

Aceste registre au lungimea de 8 biţi.

Dacă aplicaţia o cere, introducerea unui nou modul sau modificarea celor existente nu ridică nici o problemă deosebită.

Se utilizează următoarele microoperaţii, cu ajutorul cărora se vor descrie instrucţiunile şi operaţiile microprocesorului:

a) - microoperaţia de transfer :

MIT(R1,R2);

unde : R1 - registru destinaţie;
R2 - registru sursă.

b) - microoperaţia de adunare :

MIA(R1,R2,R3,R4);

unde : R1,R2,R3 - registre ale căror conţinuturi se adună;
R4 - registru ce conţine rezultatul adunării.

In definiţie s-a prezentat numărul maxim de registre care pot participa la adunare. Intr-o configuraţie simplificată, microoperaţia de adunare poate fi descrisă astfel :

MIA(R1,R2,R4);

cînd operaţia de adunare se efectuează numai între două registre (R1 şi R2). De asemenea, unul dintre registrele care participă la adunare poate fi R4, deci vom avea :

MIA(R1,R4,R4);
MIA(R1,R3,R4,R4);

c) - microoperaţia de scădere :

MIS(R1,R2,R3,R4);

unde : R1 - registru care păstrează descăzutul;
R2,R3 - registre care păstrează scăzătoarele;
R4 - registru care păstrează rezultatul.

Si aici, unul dintre registrele R2,R3 poate lipsi sau R4 să conţină, în faza iniţială, unul dintre operanzi. Deci avem variantele :

MIS(R1,R2,R4);
MIS(R4,R2,R4);
MIS(R4,R2,R3,R4);

d) - microoperaţia de rotaţie :

MIR(R1,R2,IR);

unde : R1,R2 - registre care participă la efectuarea rotaţiei;
IR - indicator ce precizează sensul rotaţiei :
IR = 1 - rotaţie dreapta;
IR = 0 - rotaţie stînga.

Unul dintre registre poate lipsi, situaţie în care rotaţia afectează numai registrul prezent. Deci microoperaţia se prezintă sub forma :

MIR(R1,IR).

Ca o exemplificare, pentru microprocesorul I8080, este vorba de efectuarea rotaţiei cu sau fără registrul de condiţie CY.

e) - microoperaţia de salt condiţionat :

MIC(RC,R);

unde : RC - registrele de condiţie;
R - registru ce conţine adresa de salt.

f) - microoperaţia inefectivă :

MIN.

Fără a avea un rol executabil, această microoperaţie se utilizează pentru a asigura o corespondenţă între execuţia simulată şi cea reală, a unei instrucţiuni.

g) - microoperaţii logice :

MLA(R1,R2,R3) - pentru realizarea funcţiei SI;
MLO(R1,R2,R3) - pentru realizarea funcţiei SAU;
MLY(R1,R2,R3) - pentru realizarea funcţiei SAU-EXCLUSIV

unde R1 şi R2 sînt registre ce conţin operanzii, iar R3 va conţine rezultatul.

Cu microoperaţiile definite se poate descrie un set larg de instrucţiuni ale circuitelor integrate LSI. Intr-o variantă mai restrînsă se poate renunţa la unele tipuri de microoperaţii, funcţia lor efectuîndu-se cu ajutorul celorlalte rămase, aceasta, desigur, în detrimentul unor descrieri mai complexe.

In cazul circuitelor integrate LSI, problema testării poate fi privită din următoarele puncte de vedere :

- stabilirea, activarea şi, implicit, verificarea modulelor ce formează circuitul LSI, astfel încît orice defect ce apare la intrare sau în configuraţia sa internă să fie transmis la o ieşire. In această fază sînt necesare, în majoritatea cazurilor, numai instrucţiunile de transfer, deci se reduce şi setul de microoperaţii, definit anterior;
- verificarea completă a circuitului integrat LSI. In această fază este necesară verificarea completă a întregului set de instrucţiuni aferente, într-o ordine prestabilită.

In majoritatea cazurilor de testare a plachetelor echipate cu circuite integrate LSI, MSI şi SSI se utilizează primul mod de lucru. Al doilea mod de lucru este destinat, în special, testării circuitelor LSI, în faza de elaborare.

Plecînd de la aceste microoperaţii, MPA de tip registru

se realizează astfel :

- intrările şi ieşirile de date (I_D, respectiv Z_D) sînt legate la magistrale;
- intrările de comandă **Icd**, sînt următoarele :
  - L - comanda de încărcare; DDS - deplasare dreapta/stînga :
    L = $\overline{MIT} \cdot MIR$; DDS = X - încărcare paralelă;
    L = $MIT \cdot \overline{MIR}$; DDS = IR - deplasare dreapta;
    L = $\overline{MIT} \cdot MIR$; DDS = $\overline{IR}$ - deplasare stînga;
    L = $\overline{MIT} \cdot \overline{MIR}$; DDS = X - conţinutul registrului nu se schimbă.
  - TC - tactul de comutare.
- ieşirile de control, cînd există, cuprind semnalizările de condiţie CY, Z, P, S, C. Aceste ieşiri sînt legate, direct, pe intrările de date ale respectivelor registre de condiţie, în care se încarcă automat, în momentul apariţiei. Registrele respective, de cîte 8 biţi, conţin în poziţia cea mai puţin semnificativă, valoarea condiţiei, iar ceilalţi biţi au valoarea egală cu zero. Aceasta este o excepţie de la modul de definire al registrelor şi s-a admis pentru a înlesni prelucrările, în sensul existenţei unui mod de lucru universal şi cînd se operează asupra condiţiilor.

Cu aceste precizări, MPA de tip registru se realizează după procedeul prezentat în paragraful anterior.

Utilizînd aceste microoperaţii, instrucţiunile se descriu ca succesiuni finite de microoperaţii. De exemplu, pentru microprocesorul I8080 avem :

- instrucţiunea NOP se descrie astfel :

```
MIT(RADR,NADR);
MIA(NADR,B1,NADR);
MIT(RI,RERED).
```

- instrucţiunea MOV r1,r2 se descrie astfel :

```
MIT(RADR,NADR);
MIA(NADR,B1,NADR);
MIT(RI,RERED);
MIT(RET,r2);
MIT(r1,RET).
```

Deoarece rolul unităţii de comandă şi control este preluat de către utilizator, microoperaţia MIT(RI,RERED) nu are nici un efect, deci se poate înlocui cu microoperaţia MIN. In această situaţie, instrucţiunea NOP se descrie astfel :

```
MIT(RADR,NADR);
```

MIA(NADR,B1,NADR);
MIN.

Un alt rol al microoperaţiei MIN este cel de descriere, dacă se doreşte, a stărilor maşină în care nu se execută, în mod explicit, nici o microoperaţie. De exemplu, instrucţiunea MOV M,r a microprocesorului I8080 nu execută, explicit, nici o microoperaţie în starea T5 a ciclului maşină M1. In acest caz, utilizînd microoperaţia MIN, instrucţiunea se descrie astfel :

MIT(RADR,NADR)
MIA(NADR,B1,NADR)
MIN
MIT(RET,r)
MIN
MIT(RADR,HL)
MIT(REREG,RET).

Registrele care au acces direct la magistralele externe sînt REREG şi RADR.

Comportarea modulelor senzoriale se verifică gradat, activînd calea de la respectivul modul, la unul din modulele cu acces direct la magistralele externe. Calea de activare se va realiza cît mai scurtă şi să nu conţină alte module neverificate. Aceasta impune utilizarea, mai întîi, a acelor instrucţiuni care implică un număr minim de module, dacă este posibil, numai acela care se verifică. Modulele verificate şi declarate corecte vor fi utilizate pentru verificarea altora. Pentru a ţine cont de aceste elemente este necesară stabilirea unui algoritm al succesiunii în execuţie a instrucţiunilor.

In vederea construirii algoritmului se apelează la graful microprocesorului. Modulele din configuraţia microprocesorului constituie nodurile grafului, iar microoperaţiile de transfer descriu liniile grafului. In fig.2.15 se prezintă graful microprocesorului I8080.

O primă fază în construcţia algoritmului o constituie notarea liniilor grafului. Plecînd de la notaţia generală L n.m.p, a unei linii, regulile de definire a valorilor indicilor n,m şi p sînt următoarele :

a) n = 1 - pentru liniile de la/la modulele REREG şi RADR;
   m = 1 - pentru liniile de la modulele REREG şi RADR, la celelalte module;
   m = 2 - pentru liniile de la celelalte module, la modulele REREG şi RADR;

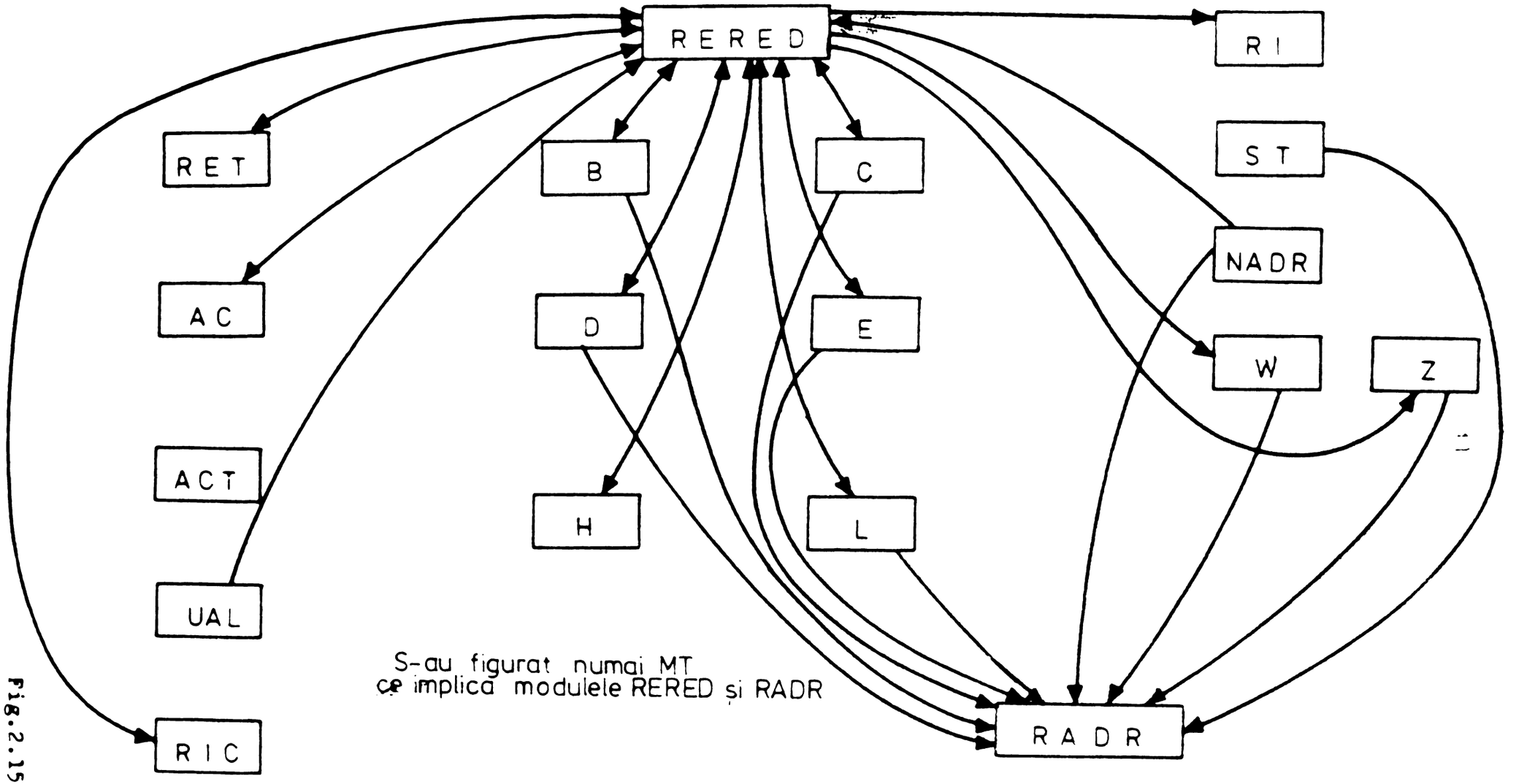
p = 0 - pentru liniile aferente microoperaţiilor (MT), care transferă informaţii de la alte module, la modulele RERED sau RADR. Aceste informaţii sînt caracterizate astfel :
- nu au fost prelucrate şi nici nu urmează a fi prelucrate, în instrucţiunea din care face parte MT;
- nu provin din modulele RERED sau RADR, în urma unui transfer executat în instrucţiunea din care face parte MT;

p = 1 - pentru liniile aferente microoperaţiilor (MT), care transferă informaţii de la alte module, la modulele RERED sau RADR. Aceste informaţii sînt caracterizate astfel :
- au fost prelucrate înainte de transfer în instrucţiunea din care face parte MT;
- nu provin din modulele RERED sau RADR, în urma unui transfer executat în instrucţiunea din care face parte MT;

p = 2 - liniile aferente microoperaţiilor (MT), care transferă informaţii de la alte module, la modulele RERED sau RADR. Aceste informaţii sînt caracterizate astfel :
- nu au fost prelucrate înainte de transfer, dar urmează a fi prelucrate, după transfer, de alte microoperaţii din instrucţiunea din care face parte MT;
- nu provin din modulele RERED sau RADR, în urma unui transfer executat în instrucţiunea din care face parte MT;

p = 3 - pentru liniile aferente microoperaţiilor (MT), care transferă informaţii de la alte module, la modulele RERED sau RADR. Aceste informaţii sînt caracterizate astfel :
- nu au fost prelucrate înainte de transfer, dar urmează a fi prelucrate, după transfer, de alte microoperaţii din instrucţiunea din care face parte MT;
- provin din modulele RERED sau RADR, în urma unui transfer executat în instrucţiunea din care face parte MT;

p = 4 - pentru liniile aferente microoperaţiilor (MT), care



S-au figurat numai MT
ce implica modulele RERED și RADR

Fig.2.15.

transferă informaţii de la alte module, la modulele RERED sau RADR. Aceste informaţii sînt caracterizate astfel :

- au fost prelucrate, înainte de transfer, în instrucţiunea din care face parte MT;
- provin din modulele RERED sau RADR, în urma unui transfer executat în instrucţiunea din care face parte MT.

Liniile aferente microoperaţiilor care execută incrementări sau decrementări cu o unitate se notează cu L123, respectiv L124, în funcţie de elementul în care se execută.

b) n = 2 - pentru liniile, care realizează legături între modulele, care au, ambele, legături de tipul L123 sau L124;

c) n = 3 - pentru liniile, care realizează legături între modulele, dintre care, unul singur are legături de tipul L123 sau L124;

d) n = 4 - pentru liniile, care realizează legături între modulele, dintre care, nici unul nu are legături de tipul L123 sau L124.

In cazurile b, c şi d nu interesează valorile indicilor m şi P. Se observă că regulile de notare sînt influenţate, direct, de posibilităţile de transfer între modulul în cauză şi modulele ce au acces direct la cel puţin una din magistralele externe ale circuitului LSI.

Utilizînd aceste notaţii, în cazul microprocesorului I8080, instrucţiunea MVI r, data realizează următoarele legături :

MIT(RADR,NADR) - L123;
MIA(NADR,B1,NADR) - L124;
MIN
MIN
MIN
MIT(RADR,NADR) - L124;
MIN
MIT(r,RERED) - L11;

Instrucţiunea ORI data realizează următoarele legături :

MIT(RADR,NADR) - L123;
MIA(NADR,B1,NADR) - L124;
MIN

MIT(ACT,AC) - L3;
MIN
MIT(RADR,NADR) - L124;
MIA(NADR,B1,NADR) - L123;
MIT(RET,RERED) - L11;
MIN
MLO(ACT,RET,UAL)
MIT(AC,UAL) - L2.

Cînd se utilizează notaţia "X" nu interesează valoarea indicelui.

Cu acestea, paşii algoritmului, care stabileşte ordinea în execuţie a instrucţiunilor, sînt următorii :

a) - se execută, mai întîi, instrucţiunile descrise prin numărul cel mai mic de microoperaţii şi conţin numărul minim de legături, de tipul L11 şi L124;
b) - se execută instrucţiunile, rămase, care conţin numai legături de tipul L11 şi L124;
c) - se execută instrucţiunile, care conţin numai legături de tipul L11, L123 şi L124, executate în această ordine;
d) - se execută instrucţiunile, care conţin numai legături de tipul L11, L124, L123, executate în această ordine. Aceste instrucţiuni se execută de două ori, iar rezultatul se interpretează după a doua execuţie;
e) - se execută instrucţiunile, care conţin numai legături de tipul L11, L123, L124, L12o, L121;
f) - se execută instrucţiunile, care conţin legături de tipul L11, L12X şi L2, începînd cu cele care conţin numărul cel mai mic de legături de tipul L2;
g) - se execută instrucţiunile, care conţin legături de tipul L11, L12X şi L3, începînd cu cele care conţin numărul cel mai mic de legături de tipul L3;
h) - se execută instrucţiunile, care conţin legături de tipul L11, L12X şi L4, începînd cu cele care conţin numărul cel mai mic de legături de tipul L4;
i) - se execută instrucţiunile rămase, în ordinea complexităţii, stabilită în funcţie de numărul de legături de tipul L2, L3, L4, legăturile de tipul L2 fiind considerate cele mai simple iar L4, cele mai complicate, din punctul de vedere al operaţiei de testare.

Rezultatul ordonării în execuţie a instrucţiunilor este stabilirea succesiunii de verificare a modulelor.

In continuare, pentru testarea modulelor se utilizează o metodă convenţională.

Datorită numărului mare de informaţii memorate, problema se poate pune şi altfel : să se stabilească setul de instrucţiuni, care pot asigura o verificare a unui modul dat, indiferent de starea celorlalte module. Si în acest caz se poate stabili un algoritm de tipul celui enunţat.

Din punctul de vedere al utilizatorului, se cer următoarele informaţii :

- stabilirea modulelor, care au acces direct la magistralele externe ale circuitului LSI;
- descrierea setului de instrucţiuni cu ajutorul microoperaţiilor.

Programul de testare automată va realiza :

- utilizînd modulele din configuraţia circuitului LSI, pe care le stabileşte din analiza microoperaţiilor, execută codificarea legăturilor şi stabileşte ordinea în execuţie a instrucţiunilor;
- pentru verificarea fiecărui modul cu ajutorul instrucţiunilor alocate, apelează programul de stabilire a vectorilor de testare, care lucrează pe baza unei metode convenţionale;
- interpretează rezultatul pentru fiecare modul în parte şi ia decizia de continuare sau nu a operaţiei de testare.

In vederea obţinerii vectorilor de testare, fiecare modul se descrie sub forma MPA. Astfel se asigură o descriere unitară pentru toate tipurile de circuite LSI, MSI şi SSI, cu care poate fi echipată o plachetă.

Avantajele cele mai importante ale acestui mod de descriere sînt simplitatea, economia de memorie şi posibilitatea utilizării aceleiaşi metode pentru stabilirea vectorilor de testare, în cazul plachetelor echipate cu orice tip de circuite logice. Un algoritm destinat acestui scop se prezintă în paragraful următor.

Totodată, descrierea sub forma MPA conţine şi secvenţele de combinaţii ce pot fi utilizate pentru iniţializarea circuitului respectiv, indiferent de starea în care se află.

2.3. Algoritm pentru elaborarea vectorilor de testare

Procesul de stabilire a vectorilor de testare implică considerarea a două scheme logice : cea corectă, C, şi cea defectă, F.

Dacă $X = (X_1X_2...X_n)$ reprezintă vectorul variabilelor ce descriu semnalele, care se aplică pe intrările schemei logice, a cărei stare este caracterizată de vectorul $Y^* = (Y_1^*, Y_2^*...Y_n^*)$, pentru detectarea unui defect este necesar ca cel puţin o ieşire să realizeze relaţia :

$$Z_i^F \oplus Z_i^C = 1 \; ; \qquad (2.3)$$

unde :
$$Z_i^F = f^F(X_1, X_2, ...X_n, Y_1^*, Y_2^*, ...Y_P^*)$$
$$Z_i^C = f^C(X_1, X_2, ...X_n, Y_1^*, Y_2^*, ...Y_P^*) \qquad (2.4)$$

reprezintă funcţiile realizate de schema defectă, respectiv corectă, la ieşirea Z_i.

Vectorul $(X_1^1, X_2^1, ...X_n^1)$ reprezintă un set de valori, care aplicate pe intrările schemei logice determină, la ieşire, valoarea logică "1".

Vectorul $(X_1^0, X_2^0, ...X_n^0)$ reprezintă un set de valori, care aplicate pe intrările schemei logice determină, la ieşire, valoarea logică "0".

Algoritmul urmăreşte activarea cel puţin a unei căi de la o intrare, la ieşire. In acest fel se asigură ca de la fiecare intrare să se controleze valoarea ieşirii, controlul efectuîndu-se pe calea activată. Orice modificare, în comportare, ce apare pe calea activată, trebuie să fie vizibilă la ieşire.

Pentru ca o intrare să controleze valoarea ieşirii, este necesar ca aceasta să se modifice dacă valoarea logică a semnalului aplicat pe intrarea respectivă se modifică, iar semnalele de pe toate celelalte intrări rămîn neschimbate. Deci, dacă se analizează intrarea caracterizată prin variabila logică X_i, se caută combinaţii de semnale în care să difere numai valoarea variabilei X_i şi care, aplicate pe intrări, să determine valori diferite, la ieşire. Aceste combinaţii se găsesc în setul de soluţii al ecuaţiilor :

$$f^C(X_1, X_2, ...X_n, Y_1^*, Y_2^*, ...Y_P^*) = 0 \qquad (2.5)$$

$$f^C(X_1, X_2, ...X_n, Y_1^*, Y_2^*, ...Y_P^*) = 1 \qquad (2.6)$$

Pentru a reduce numărul de calcule, este necesar să se rezolve acea ecuaţie, care are numărul minim de soluţii.

Cu aceste precizări, pașii algoritmului sînt următorii :

a) - funcția $f^C(X_1,X_2,...X_n, Y_1^*,Y_2^*,...Y_P^*)$ se aduce sub forma de sumă de produse, care conțin variabile adevărate sau negate;

b) - se determină, care din ecuațiile (2.5),respectiv (2.6) are numărul minim de soluții și se rezolvă acea ecuație. Rezultă m soluții de forma : $(X_1,X_2,...X_i,...X_n, Y_1^*,Y_2^*...Y_P^*)$.
$K = 1; i = 1;$

c) - pentru ca soluția K, din setul de m soluții, să activeze calea de la intrarea notată prin variabila X_i, la ieșire, este necesar ca setul de m soluții să nu conțină combinația :
$(X_1,X_2,...\bar{X}_i,...X_n, Y_1^*,Y_2^*,...Y_P^*)$.

Dacă răspunsul este afirmativ, atunci combinația de mai sus este un vector de testare, care permite verificarea comportării căii de la intrarea notată cu variabila X_i, la ieșire. Intrarea respectivă se notează, în continuare, cu P.

Dacă răspunsul este negativ, combinația respectivă nu constituie un vector de testare pentru calea de la intrarea X_i, la ieșire.

d) - K = K+1; dacă $K \leq m$, se reia cu pasul C; în caz contrar se trece la pasul e ;

e) - i = i+1; dacă $i \leq (n+P)$, atunci k = 1 și se reia cu pasul C. In caz contrar STOP.

In final rezultă cel puțin un vector de testare pentru fiecare cale de la o intrare, la ieșire. Totodată, se precizează faptul că un vector poate activa mai multe căi simultan. Această ultimă problemă va fi tratată pe un exemplu.

Cele prezentate sînt valabile și în cazul în care se urmărește activarea unei căi ce va fi controlată de către o variabilă de stare, Y_i^*. In acest caz raționamentul se face pentru variabila Y_i^*.

Modul de rezolvare a punctelor a și b este esențial în reducerea numărului de operații cerut de algoritm. Pentru aceasta este necesar să se rezolve acea ecuație (2.5) sau (2.6), care are numărul minim de soluții. Deci, trebuie aplicat un algoritm cu ajutorul căruia să se stabilească ecuația cu număr minim de soluții, înainte de rezolvare. In acest sens funcția se aduce la forma de sumă de produse, fiecare produs conținînd variabilele adevărate sau negate. Se notează cu U_i, fiecare produs al sumei, iar

cu V_j (unde V_j poate fi o variabilă de intrare Y_j sau o variabilă ce caracterizează o stare, Y_j^*), o variabilă a unui produs. Cu aceste notaţii funcţia se poate scrie sub forma :

$$f^C(X_1,X_2,\ldots X_n,\ Y_1^*,Y_2^*,\ldots Y_P^*) = \sum_{i=1}^{m} U_i. \qquad (2.7)$$

In continuare, pentru a stabili ecuaţia cu număr minim de soluţii, se transformă fiecare produs al sumei astfel încît rezolvarea uneia din ecuaţiile (2.5) sau (2.6) să nu introducă soluţii multiple. Acest lucru se obţine dacă fiecare produs diferă de oricare alt produs al sumei, cel puţin prin valoarea unei variabile. După ce s-au transformat produsele astfel încît să respecte regula enunţată mai sus, se poate determina numărul de soluţii pentru oricare din ecuaţiile (2.5) sau (2.6). Dacă ne referim la ecuaţia (2.6), fiecare produs transformat introduce 2^{n-K} soluţii, unde n reprezintă numărul de variabile ale funcţiei iar K, numărul de variabile conţinute în produsul respectiv. Aplicînd acest rationament, la fiecare produs, rezultă, în total, n_S soluţii. Dacă este îndeplinită relaţia $n_S \leq 2^{n/2}$ se va rezolva ecuaţia (2.6). In caz contrar se va rezolva ecuaţia (2.5).

Schema logică a acestui algoritm se prezintă în fig.2.16.

Se compară fiecare produs cu toate celelalte produse ale sumei. In urma comparaţiei se iau următoarele decizii :

a) - dacă în urma comparaţiei produsului U_i cu produsul U_j, $j>i$, rezultă că diferă cel puţin prin valoarea unei variabile, asupra produsului U_i nu se aplică nici o transformare. Exemplu : $V_1\overline{V}_2 + V_2V_3$;

b) - dacă produsul U_i diferă de reuniunea U_iUU_j, $j>i$, numai printr-o variabilă V_1, atunci se aplică transformarea :

$$U_i = U_iV_1 + U_i\overline{V}_1$$

Termenul U_iV_1 va fi absorbit de produsul U_j. Exemplu :

$$\overline{V}_1\overline{V}_3 + V_2\overline{V}_3 = \overline{V}_1\overline{V}_2\overline{V}_3 + \overline{V}_1V_2\overline{V}_3 + V_2\overline{V}_3 = \overline{V}_1\overline{V}_2\overline{V}_3 + V_2\overline{V}_3;$$

c) - dacă produsul U_i diferă de reuniunea U_iUU_j $j>i$, prin mai mult de o singură variabilă, se aplică transformarea :

$$U_i = U_iV_1 + U_i\overline{V}_1 \quad ;$$

în care, V_1 este prima variabilă conţinută de produsul U_j, dar care nu se află în produsul U_i. In felul acesta s-a introdus un termen suplimentar în sumă (U_iV_1), care va fi pre-

lucrat după aceleaşi reguli ca şi produsele iniţiale.

Exemplu : $V_2\bar{V}_3 + \bar{V}_1\bar{V}_4 = V_1V_2\bar{V}_3 + \bar{V}_1\bar{V}_4 + \bar{V}_1V_2\bar{V}_3$

In continuare se prezintă acest algoritm pe un exemplu. Fără a pierde din generalitate, se utilizează numai variabile ce caracterizează semnalele aplicate pe intrări, X_i.

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_3 + X_2\bar{X}_3 + \bar{X}_1\bar{X}_4 + X_2\bar{X}_4 \; ; \qquad (2.8)$$

(b)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3 + X_2\bar{X}_3 + \bar{X}_1\bar{X}_4 + X_2\bar{X}_4 \; ;$$

(b)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_2\bar{X}_3 + \bar{X}_1\bar{X}_4 + X_2\bar{X}_4 \; ;$$

(c)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_1X_2\bar{X}_3 + \bar{X}_1\bar{X}_4 + X_2\bar{X}_4 + \bar{X}_1X_2\bar{X}_3 \; ;$$

(b)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_1X_2\bar{X}_3X_4 + \bar{X}_1\bar{X}_4 + X_2\bar{X}_4 + \bar{X}_1X_2\bar{X}_3 \; ;$$

(b)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_1X_2\bar{X}_3X_4 + \bar{X}_1\bar{X}_2\bar{X}_4 + X_2\bar{X}_4 + \bar{X}_1X_2\bar{X}_3 \; ;$$

(c)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_1X_2\bar{X}_3X_4 + \bar{X}_1\bar{X}_2\bar{X}_4 + X_1X_2\bar{X}_4 + \bar{X}_1X_2\bar{X}_3 + \bar{X}_1X_2\bar{X}_4 \; ;$$

(b)

$$f(X_1,X_2,X_3,X_4) = \bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_1X_2\bar{X}_3X_4 + \bar{X}_1\bar{X}_2\bar{X}_4 + X_1X_2\bar{X}_4 + \bar{X}_1X_2\bar{X}_3X_4 + \bar{X}_1X_2\bar{X}_4 .$$

Se observă şi din exemplul prezentat, că modul de lucru este simplu şi destul de rapid. Un produs odată analizat şi eventual transformat, nu mai este luat în considerare. În felul acesta se economiseşte timp şi capacitate de memorare.

Cu ajutorul ultimei forme de prezentare a funcţiei f se poate determina numărul minim de soluţii astfel :

$$f(X_1,X_2,X_3,X_4) = 1 \; ; \qquad (2.9)$$

$$\bar{X}_1\bar{X}_2\bar{X}_3X_4 + X_1X_2\bar{X}_3X_4 + \bar{X}_1\bar{X}_2\bar{X}_4 + X_1X_2\bar{X}_4 + \bar{X}_1X_2\bar{X}_3X_4 + \bar{X}_1X_2\bar{X}_4 = 1 \; ;$$

$$n_S = 2^{4-4} + 2^{4-4} + 2^{4-3} + 2^{4-5} + 2^{4-4} + 2^{4-3} = 9 .$$

Deci $n_S = 9 > 2^4/2$ şi în consecinţă, numărul minim de soluţii va fi obţinut prin rezolvarea ecuaţiei :

$$f(X_1,X_2,X_3,X_4) = 0 \; ; \qquad (2\text{-}1o)$$

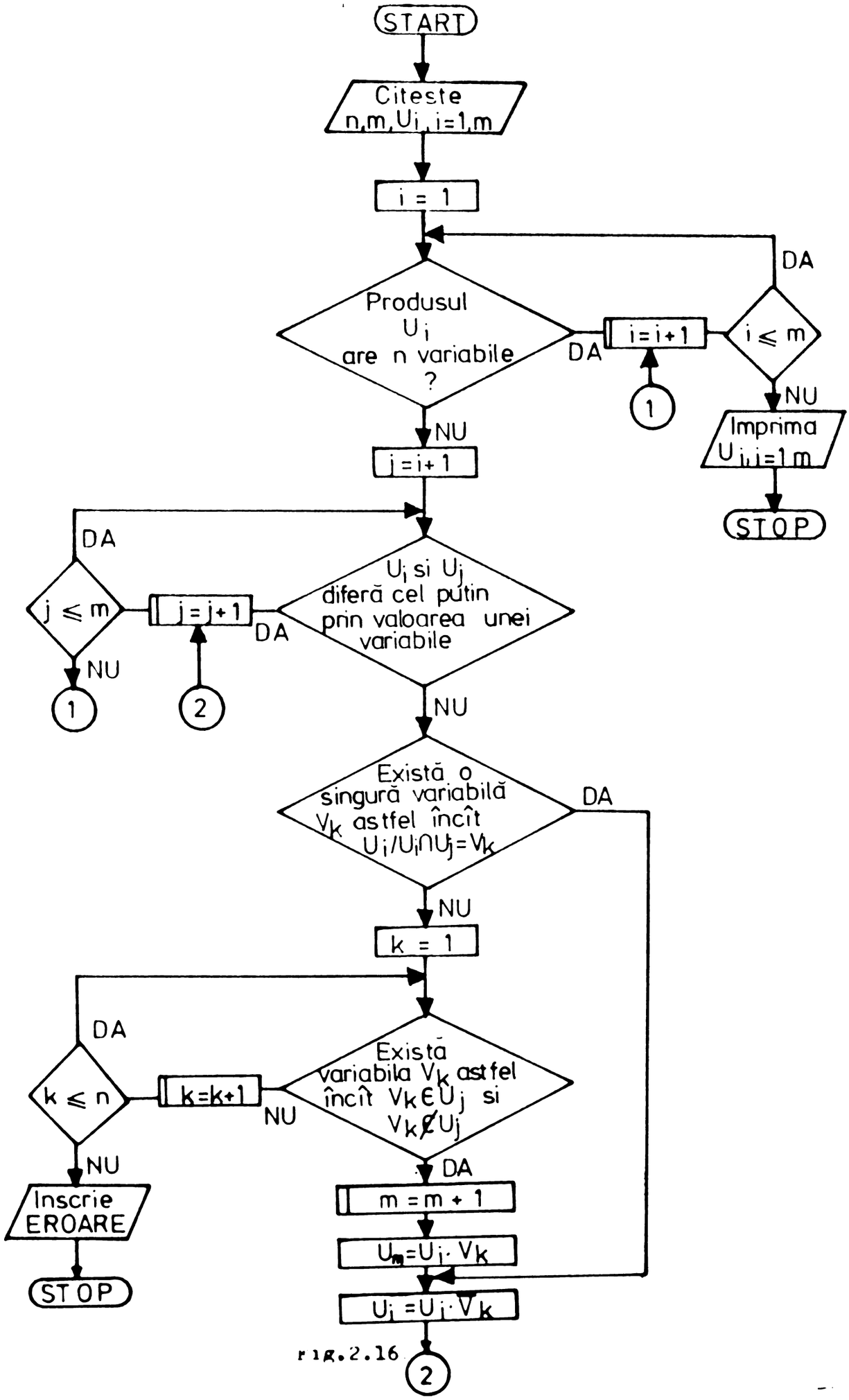

START
Citeste n,m,Ui, i=1,m
i = 1
Produsul Ui are n variabile ?
DA
i=i+1
i ≤ m
DA
NU
1
Imprima Ui, i=1,m
STOP
NU
j=i+1
DA
j ≤ m
j=j+1
Ui si Uj diferă cel puțin prin valoarea unei variabile
DA
NU
1
2
NU
Există o singură variabilă Vk astfel încît Ui/Ui∩Uj=Vk
DA
NU
k = 1
DA
k ≤ n
k=k+1
Există variabila Vk astfel încît Vk∈Uj si Vk∉Uj
NU
DA
NU
Inscrie EROARE
STOP
m = m + 1
Um=Uj·Vk
Ui =Ui·Vk
2

Fig.2.16

Cu ajutorul soluţiilor astfel obţinute se determină vectorii de testare după metodologia prezentată anterior. Dintre vectorii de testare, care activează aceeaşi cale, se reţine unul singur, ceilalţi se elimină. Pentru aceasta se aplică un procedeu de minimizare grafo-analitică, care constă în construcţia tabelului vectorilor de testare, TAVT. Acest tabel conţine, pe fiecare coloană, calea activată, iar pe linii, vectorii de testare. Dacă o cale C_i este activată de către vectorul de testare VT_j, atunci la intersecţia coloanei C_i cu linia VT_j se introduce cifra "1". In caz contrar se introduce cifra "0".

Pentru minimizarea numărului vectorilor de testare se execută următoarele operaţii :

- se elimină acele rînduri, care din punct de vedere al cifrei "1" sînt conţinute în alte rînduri. Rămîne un singur rînd, care le poate acoperi pe toate celelalte eliminate;
- se elimină acele coloane, care din punctul de vedere al cifrei "1" sînt conţinute în alte coloane. Rămîne numai coloana, care poate acoperi toate coloanele eliminate.

In continuare se exemplifică modul de construcţie al vectorilor de testare, atît pentru cazul schemelor combinaţionale, cît şi al celor secvenţiale.

a) Se analizează schema din fig.2.17.

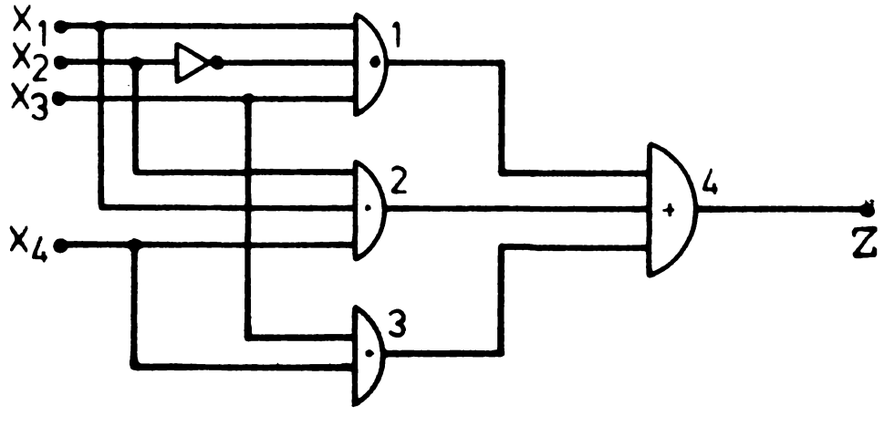


Fig.2.17.

Funcţia realizată la ieşirea Z este următoarea :

$$Z(X_1,X_2,X_3,X_4) = X_1\bar{X}_2X_3 + X_1X_2X_4 + X_3X_4 \; ; \qquad (2.11)$$

Aplicînd algoritmul de transformare se obţine următoarea formă :

$$Z(X_1,X_2,X_3,X_4) = X_1\bar{X}_2X_3\bar{X}_4 + X_1X_2\bar{X}_3X_4 + X_3X_4 \; ; \qquad (2.12)$$

Pentru ecuaţia :

$$Z(X_1,X_2,X_3,X_4) = 1 \qquad (2.13)$$

rezultă $n_S = 2^{4-4} + 2^{4-4} + 2^{4-2} = 6 < 2^4/2$, deci se va rezolva ecuaţia (2.13). Soluţiile se prezintă în tabelul din fig.2.18.

k	X1	X2	X3	X4
1	0	0	1	1
2	0	1	1	1
3	1	0	1	0
4	1	0	1	1
5	1	1	0	1
6	1	1	1	1

Fig.2.18.

Vectorii de testare, care activează cîte o singură cale, sînt următorii :

- pentru activarea căii C_1 : X_1 - poarta 1 - Z — VT_1 = P01O ;
- pentru activarea căii C_2 : X_1 - poarta 2 - Z — VT_2 = P1O1 ;
- pentru activarea căii C_3 : X_2 - poarta 1 - Z — VT_3 = 1P1O ;
- pentru activarea căii C_4 : X_2 - poarta 2 - Z — VT_4 = 1P01 ;
- pentru activarea căii C_5 : X_3 - poarta 1 - Z — VT_5 = 10P0 ;
- pentru activarea căii C_6 : X_3 - poarta 3 - Z — VT_6 = 00P1 ;

  VT_7 = 01P1 ;
- pentru activarea căii C_7 : X_4 - poarta 2 - Z — VT_8 = 110P ;
- pentru activarea căii C_8 : X_4 - poarta 3 - Z — VT_9 = 001P ;

  VT_{10} = 011P .

Vectorii de testare care activează căi multiple sînt următorii :

- pentru activarea căilor : C_5 : X_3 - poarta 1 - Z
  C_6 : X_3 - poarta 3 - Z --- VT_{11} = 10P1;
- pentru activarea căilor : C_7 : X_4 - poarta 2 - Z
  C_8 : X_4 - poarta 3 - Z --- VT_{12} = 111P.

Configuraţia TAVT se prezintă în fig.2.19.

C / VT	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
VT_1	1							
VT_2		1						
VT_3			1					
VT_4				1				
VT_5					1			
VT_6						1		
VT_7						1		
VT_8							1	
VT_9								1
VT_{10}								1
VT_{11}					1	1		
VT_{12}							1	1

Fig.2.19.

In urma analizei TAVT se poate reţine numărul minim al vectorilor de testare, astfel ca să se asigure activarea fiecărei căi.

Astfel setul de vectori VT_1, VT_2, VT_3, VT_4, VT_{11} şi VT_{12} realizează activarea tuturor căilor, dar căile C_5 şi C_6, respectiv C_7 şi C_8 sînt activate simultan. Setul de vectori VT_1, VT_2, VT_3, VT_4, VT_5, VT_6, VT_8 şi VT_9 permite activarea individuală a fiecărei căi. Se observă că în ambele cazuri s-a obţinut o reducere importantă a numărului de vectori.

Obs. Algoritmul asigură activarea tuturor căilor numai dacă schema logică nu este redundantă. De exemplu, dacă se realizează schema descrisă de funcţia :

$$Z(X_1,X_2,X_3,X_4) = X_1\bar{X}_2X_3 + X_1X_2X_4 + X_1X_3X_4 \; ; \qquad (2.14)$$

adică apare X_1 redundant în termenul al treilea, setul vec-

torilor de testare rămîne acelaşi, deci nu se construieşte vector pentru activarea căii X_1 - poarta 3 - Z.

b) Se analizează schema din fig.2.2o.

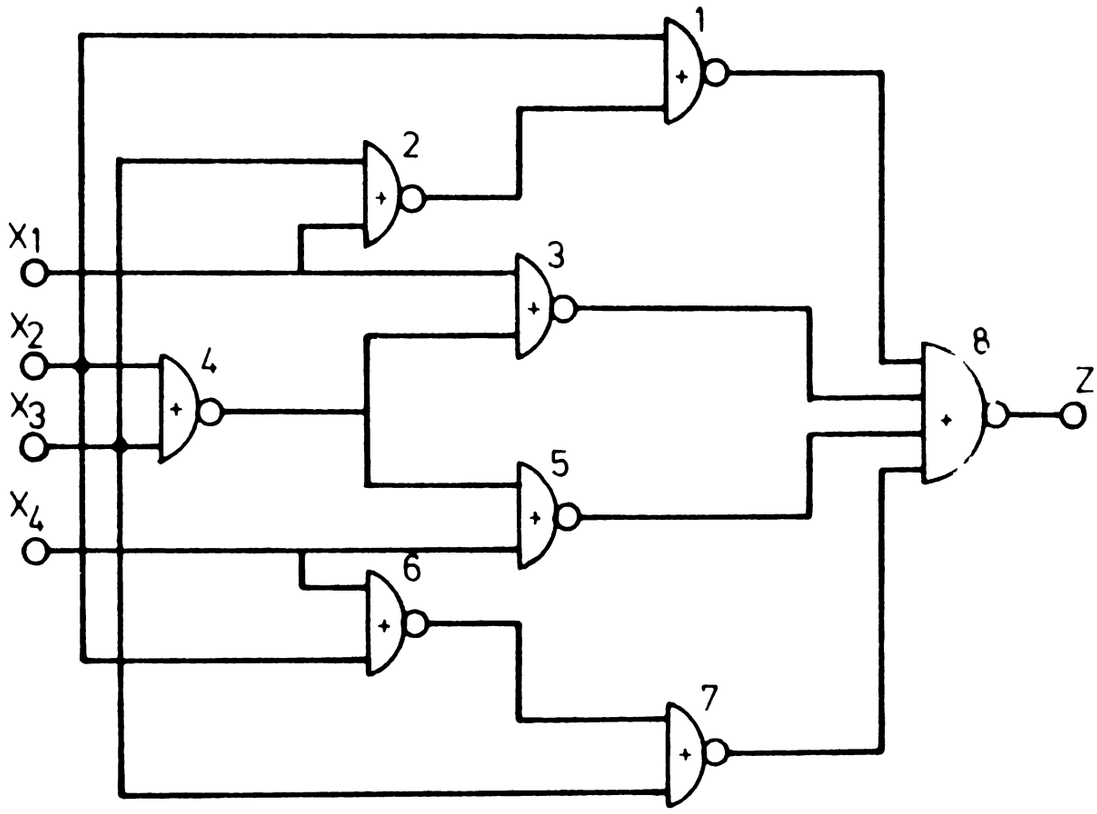


Fig.2.2o.

Funcţia realizată la ieşirea Z este următoarea :

$$Z(X_1,X_2,X_3,X_4) = X_1X_2X_3X_4 + \bar{X}_1\bar{X}_2\bar{X}_3\bar{X}_4 \ . \qquad (2.15)$$

Funcţia îndeplineşte condiţiile astfel încît să se poată stabili ecuaţia cu numărul minim de soluţii. Deci ecuaţia :

$$Z(X_1,X_2,X_3,X_4) = 1 \ ; \qquad (2.16)$$

are $n_2 = 2 < 2^4/2$ soluţii şi anume :

$$X_1 = X_2 = X_3 = X_4 = 0 \ ;$$

$$X_1 = X_2 = X_3 = X_4 = 1 \ .$$

Vectorii care activează cîte o singură cale sînt următorii :

- pentru activarea căii : $C_1 : X_1 - Y_2 - Y_1 - Z$ — $VT_1 = P000$;
- pentru activarea căii : $C_2 : X_1 - Y_3 - Z$ — $VT_2 = P111$;
- pentru activarea căii : $C_3 : X_2 - Y_1 - Z$ — $VT_3 = 1P11$;

- pentru activarea căii : C_4 : $X_3 - Y_7 - Z$ — VT_4 = 11P1;
- pentru activarea căii : C_5 : $X_4 - Y_6 - Y_7 - Z$ — VT_5 = 000P;
- pentru activarea căii : C_6 : $X_4 - Y_5 - Z$ — VT_6 = 111P.

Vectorii care activează căi multiple sînt următorii :

- pentru activarea căilor : C_7 : $X_2 - Y_4 - Y_3 - Z$
  C_8 : $X_2 - Y_4 - Y_5 - Z$ — VT_7 = 0P00;
  C_9 : $X_2 - Y_6 - Y_7 - Z$
- pentru activarea căilor : C_{10}: $X_3 - Y_4 - Y_3 - Z$
  C_{11}: $X_3 - Y_4 - Y_5 - Z$ — VT_8 = 00P0.
  C_{12}: $X_3 - Y_2 - Y_1 - Z$

Se observă că în acest exemplu există căi care nu pot fi activate individual, pentru nici un vector de testare.

In cazul activării de căi multiple, pentru a putea transmite variabila P sau $\overline{P}$ la o ieşire, se admite efectuarea operaţiilor :

$$P + \overline{P} = P \; ; \quad P.\overline{P} = \overline{P} \; .$$

Activarea de căi multiple poate apare, cînd o linie din schema logică atacă mai multe intrări de circuite. Acest lucru se evidenţiază şi în funcţia realizată de schemă, prin prezenţa variabilei ce descrie semnalul de pe linia în cauză, în mai mulţi termeni ai funcţiei. Deci, căutînd variabilele ce îndeplinesc această condiţie, se poate semnala posibilitatea apariţiei de căi multiplu activate.

Dacă se doreşte, pentru o mai bună detectare şi eventual localizare a defectelor, activarea tuturor căilor numai individual, se propune introducerea de puncte de control suplimentare. Bineînţeles, aceasta se va face numai în cazurile în care nu există altă posibilitate (cum ar fi vectori de testare care să realizeze activarea căilor şi individual, exemplul a), cum este situaţia în exemplul b. Punctele de control suplimentare vor fi active numai în faza de testare, în rest, li se vor atribui astfel de valori logice, încît să nu influenţeze funcţionarea schemei.

In continuare se prezintă un algoritm pentru introducerea punctelor de control suplimentare :

b1) - pentru fiecare vector de testare, care activează mai multe căi, se introduce numărul necesar de puncte de control. Re-

zultă, pentru toţi cei m vectori de testare, n puncte.
i = 1 ; K = 1 ;

b2) - se verifică dacă punctul de control i afectează rolul unuia dintre punctele de control ale vectorului K. Punctele vectorului K, pentru care răspunsul este afirmativ se elimină.

b3) - K = K + 1; dacă K$\leqslant$m, se revine la pasul b2;
In caz contrar se trece la pasul b4.

b4) - i = i + 1; dacă i$\leqslant$n, atunci k = 1 şi se revine la pasul b2.
In caz contrar STOP.

Pentru exemplul prezentat anterior rezultă necesitatea introducerii a trei puncte de control, introduse ca şi intrări suplimentare la porţile Y_2, Y_3 (sau Y_5) şi Y_6. Cu ajutorul valorilor logice aplicate pe aceste intrări se asigură, ca la un moment dat, să fie activată o singură cale de la o intrare, la ieşire.

c) Se analizează schema din fig.2.21.

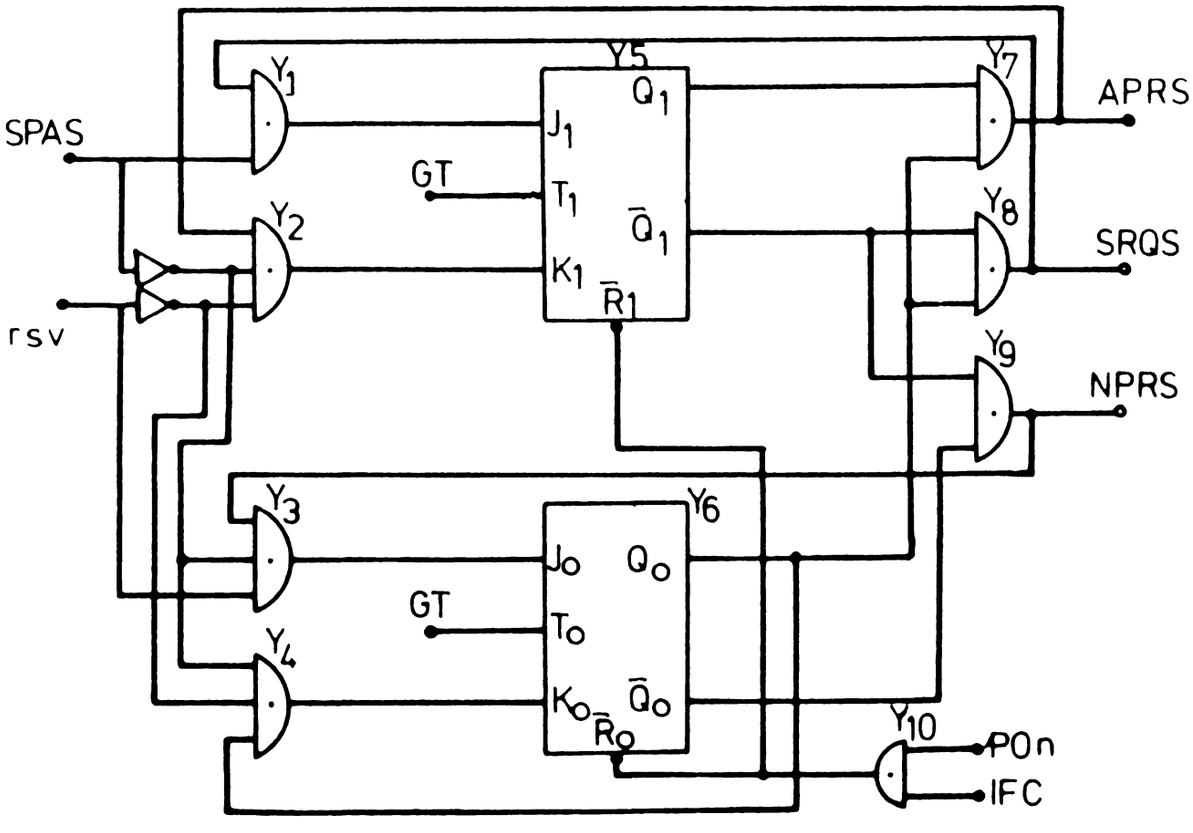


Fig.2.21.

Schema reprezintă realizarea funcţiei SR, a interfeţei universale IEEE488. După cum se observă, este o schemă secvenţială sincronă.

Se analizează numai ieşirea notată APRS. Modul de lucru în cazul celorlalte ieşiri este identic.

Valoarea semnalului APRS la momentul (t+1), în funcţie de starea la momentul t şi semnalele aplicate pe intrări, este dată de relaţia :

$$APRS_{t+1} = Q_0 \cdot SPAS + Q_0 \cdot Q_1 \cdot rsv \; ; \qquad (2.17)$$

Utilizînd notaţiile (2.4) rezultă :

$$Z(X_1, X_2, Y_1^*, Y_2^*) = Y_1^* \cdot X_1 + Y_1^* \cdot Y_2^* \cdot X_2 \; ; \qquad (2.18)$$

In urma transformării funcţiei (2.18) se obţine :

$$Z(X_1, X_2, Y_1^*, Y_2^*) = Y_1^* X_1 \bar{X}_2 + Y_1^* Y_2^* \bar{X}_1 X_2 + Y_1^* X_1 X_2 ; \qquad (2.19)$$

Deci ecuaţia :

$$Z(X_1, X_2, Y_1^*, Y_2^*) = 1 \qquad (2.2o)$$

are $n_S = 2^{4-3} + 2^{4-4} + 2^{4-3} = 5 < 2^4/2$. Soluţii.

In urma rezolvării ecuaţiei (2.2o) rezultă soluţiile prezentate în tabelul din fig.2.22.

k	rsv	SPAS	Q_0	Q_1
1	0	1	1	0
2	0	1	1	1
3	1	1	1	0
4	1	1	1	1
5	1	0	1	1

Fig.2.22.

Vectorii de testare, care activează cîte o singură cale, sînt următorii :

- pentru activarea căii C_1 : SPAS-Y_1-Y_5-Y_7-APRS — VT_1 = 0P10
  VT_2 = 1P10
- pentru activarea căii C_2 : Y_6 - Y_7 — VT_3 = 01P0
  VT_4 = 01P1
  VT_5 = 11P0
  VT_6 = 11P1
  VT_7 = 10P1
- pentru activarea căii C_3 : Y_5 - Y_7 — VT_8 = 101P

Vectorii de testare, care activează căi multiple, sînt următorii :

- pentru activarea căilor : C_4 : rsv-Y_2-Y_5-Y_7-APRS
  C_5 : rsv-Y_4-Y_6-Y_7-APRS — VT_9 = POll;
- pentru activarea căilor : C_6 : SPAS-Y_2-Y_5-Y_7-APRS
  C_7 : SPAS-Y_2-Y_5-Y_7-APRS — VT_{10}= OPll.

Dacă se reţine un singur vector din cei care activează aceeaşi cale (ceea ce se realizează în TAVT) rezultă, în final, numărul minim de vectori ce activează cîte o singură cale : VT_1, VT_3 şi VT_8.

Vectorii care activează căi multiple sînt VT_9 şi VT_{10}. Pentru căile C_4 şi C_5, respectiv C_6 şi C_7 nu există vectori de testare care să asigure activarea individuală a fiecăreia. Dacă se doreşte acest lucru se vor introduce puncte de control suplimentare.

In acest exemplu nu s-a luat în considerare activarea căii de la intrarea de tact, la ieşire. Aceasta nu ridică nici o problemă deosebită.

Obs. In cazul schemelor secvenţiale nu este, neapărat, necesară activarea explicită a căilor controlate prin variabilele de stare (căile C_2 şi C_3 din exemplul prezentat anterior), deoarece acestea se iau în considerare, implicit, cînd se activează căile de la intrările schemei, prin circuitele secvenţiale, la ieşirea schemei. Activarea explicită a căilor menţionate poate contribui, însă, la facilitatea operaţiei de localizare a defectelor.

In concluzie, se poate afirma că algoritmul prezentat permite, fără eforturi deosebite, construirea întregului set al vectorilor de testare pentru o schemă dată.

2.4. Concluzii

Problemele tratate în acest capitol permit evidenţierea următoarelor concluzii :

a) Metoda de testare prezentată asigură construirea vectorilor de testare a schemelor realizate cu circuite SSI, MSI şi LSI.

b) Circuitele componente se descriu sub forma MPA, prin

care se evidenţiază faptul că existenţa oricărui defect, ce afectează comportarea unui astfel de circuit, poate fi evidenţiată la cel puţin o ieşire.

c) Un anumit vector va permite analiza comportării unei anumite căi de la o intrare, pînă la o ieşire a schemei realizată pe plachetă. Eficienţa maximă se obţine cînd în respectiva schemă există un singur defect.

d) Vectorii de testare stabiliţi pot fi utilizaţi şi dacă în schemă există, la un moment dat, mai multe defecte, dar care nu se anihilează reciproc. Însă, în acest caz, eficienţa este mai scăzută.

e) Metoda de testare prezentată oferă informaţii minore asupra comportării legăturilor de reacţie.

Cap.3. REALIZAREA AUTOMATĂ A PROGRAMELOR DE TESTARE

3.1. Consideraţii generale

Setul de programe cu care este dotat un sistem de testare s-a perfecţionat continuu, devenind astăzi un element de bază în configuraţia sistemului şi contribuind astfel, în mare măsură, la obţinerea unor performanţe ridicate, în ceea ce priveşte detectarea şi localizarea defectelor.

Realizînd o interfaţă între utilizator şi sistemul de testare, setul de programe, închegat sub forma unui sistem de programe de testare (SPT) [N3], supraveghează, în totalitate, schimbul de informaţii cu elementul sub test, permiţînd descrierea şi utilizarea a diferite tehnici şi moduri de testare, în funcţie de specificaţiile utilizatorului şi caracteristicile elementului sub test. Se urmăreşte realizarea unei interacţiuni hard-soft performante, astfel încît utilizatorul să poată descrie caracteristicile oricărui semnal ce ar circula între sistemul de testare şi elementul sub test, iar asupra acestuia să se permită aplicarea oricărui tratament necesar şi posibil. In timp, testarea este un proces perfecţionabil, astfel că utilizatorul, pe baza rezultatelor obţinute în fazele precedente, poate modifica programele de testare, în scopul obţinerii unor performanţe finale superioare.

Factorii de care trebuie să se ţină seama la proiectarea unui SPT sînt următorii [M1],[*4],[*5],[*6],[*8],[*27],[*35] :

- concepţia unitară a tuturor programelor;
- sintaxă universală, apropiată de limbajul utilizatorilor de circuite;
- transparenţă pentru procesele interne, astfel încît vizualizarea să se poată realiza cu comenzi simple;
- concepţie şi realizare modulară;
- facilităţi de realizare şi depanare rapidă a programelor utilizatorilor sistemului de testare;
- portabilitate, cel puţin în cazul sistemelor de testare aparţinînd aceleiaşi familii;
- integrarea optimă în cadrul sistemului de operare al

echipamentului de calcul utilizat în sistemul de testare;

- posibilităţi de autotestare la nivel hard şi soft.

Eficienţa operaţiei de testare este influenţată, în mare măsură, de capabilităţile SPT.

3.2. Configuraţia sistemului de programe de testare [N3], [K2], [M1], [*4], [*5], [*8], [*17], [*20], [*24], [*32], [*35], [*46]

Se precizează faptul că organizarea şi configuraţia SPT este influenţată de tipul sistemului de testare, performanţele solicitate şi caracteristicile domeniului de aplicaţii pentru care este destinat. Componentele unui SPT se prezintă în fig.3.1.

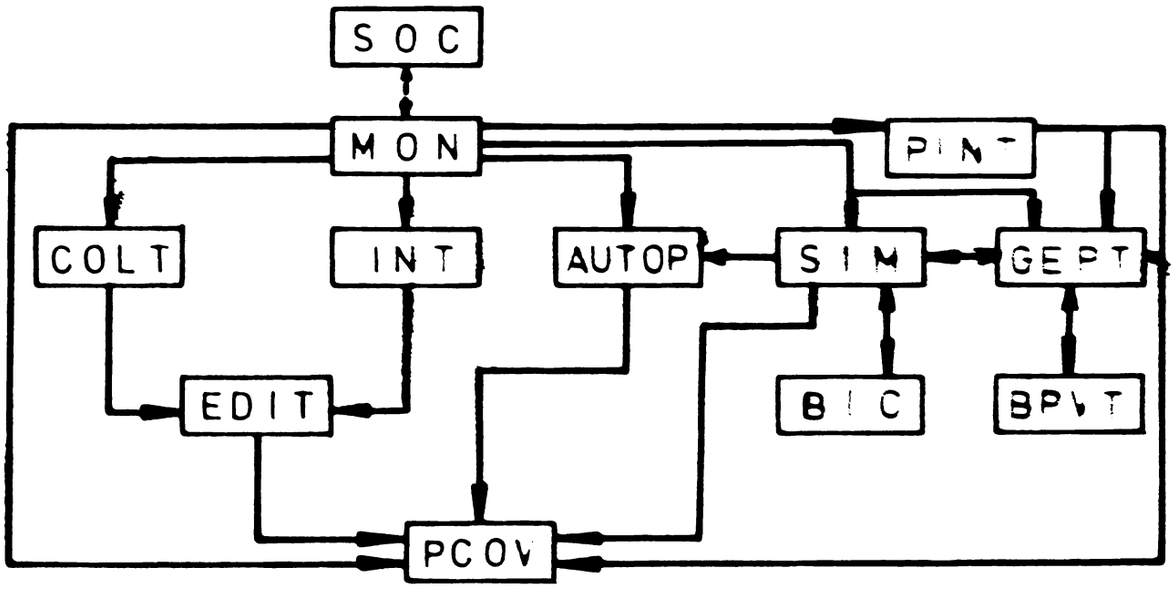


Fig.3.1.

Semnificaţia notaţiilor utilizate este următoarea :

- sistemul de operare al echipamentului de calcul, SOC;
- monitorul SPT, MON;
- compilatorul limbajului orientat spre testare, COLT;
- interpretor al limbajului orientat spre testare, INT;
- editor, EDIT;
- autoprogramator, AUTOP;
- simulator, SIM;
- generatorul programelor de testare, GEPT;

- biblioteca de circuite, BIC;
- biblioteca programelor de construire a vectorilor de testare, BPVT;
- programe pentru testarea IN-CIRCUIT, PINT;
- programe pentru conversaţia cu operatorul şi vizualizarea rezultatelor, PCOV.

Monitorul supraveghează, în totalitate, desfăşurarea operaţiei de testare, în orice regim şi mod de lucru. Conversaţia cu operatorul se realizează prin intermediul unui panou specializat sau a unei console. In acest caz se specifică :

- modul de testare : automat, pas cu pas, stop la defect, ciclare pe defect etc.;
- modul de afişare : la defect, la fiecare pas, tipul echipamentului utilizat în afişare, caracteristicile secvenţei de afişare etc.;
- comenzi utilizate în faza de testare : START, STOP, INIT etc.;
- diferite tipuri de semnalizări, legate de : rezultatul operaţiei de testare (DEFECT/CORECT), starea programului (PREGĂTIT, EXECUTIE), etc.

Tot monitorul este acela care realizează interacţiunea cu sistemul de operare al echipamentului de calcul, desfăşurîndu-şi activitatea în subordinea acestuia. In felul acesta, prin intermediul monitorului, se pot apela şi utiliza unele din capabilităţile şi resursele sistemului de operare.

Compilatorul sau interpretorul permit utilizarea unui limbaj orientat spre testare. Cerinţa de bază solicitată acestora este viteza mare de prelucrare. Dar timpul afectat operaţiei de compilare este influenţat de sintaxa limbajului. Din acest motiv trebuie admis un compromis între o sintaxă relativ complexă şi un timp de prelucrare minim.

Caracteristicile solicitate limbajelor de testare sînt [*4],[*5],[*8],[*17] :

- accesibilitatea;
- să permită utilizarea de algoritmi de testare cît mai complexi;
- să permită definirea şi utilizarea de simboluri, variabile, şiruri de caractere, subprograme;
- să dispună de un metalimbaj adecvat;
- să utilizeze o sintaxă simplă, cît mai universală.

Existenţa acestor elemente permite utilizarea tehnicii de programare structurată, prin care se asigură întocmirea programelor de testare în mod ierarhic, sub forma de unităţi funcţionale

independente, dar fiecare unitate fiind capabilă să interacţioneze cu altele, prin parametrii sau variabilele ce se transferă între unităţi. Avantajele acestui mod de lucru sînt :

- orice unitate poate fi verificată individual;
- descrierea modulară;
- reducerea timpului de întocmire a programelor;
- reducerea timpilor de punere în funcţiune şi de exploatare a programelor;
- asigurarea unui control eficient al resurselor.

Pentru descrierea şi utilizarea resurselor, a capabilităţilor sistemului automat de testare, se utilizează următoarele tehnici [M1],[*4],[*5],[*6],[*12],[*17] :

- controlul direct al resursei de către utilizator. In acest caz toate operaţiile de definire şi alocare se realizează în faza de compilare. Deoarece nu se permite utilizarea de variabile, simboluri, subprograme, utilizatorul trebuie să memoreze toate definirile şi alocările făcute, astfel încît să existe un control eficient, în orice moment, asupra întregului sistem. Limbajele ce posedă aceste caracteristici au configuraţia cea mai simplă;
- controlul orientat spre elementul sub test. In acest caz, utilizatorul poate ignora resursele, deoarece există posibilitatea definirii directe, prin alocare dinamică, a modului de testare, tipul semnalelor cu care se lucrează şi măsurătorile ce se efectuează, fără a specifica aparatele ce sînt implicate în aceste acţiuni, fiindcă se alocă şi se gestionează automat. Utilizatorul se concentrează numai asupra problemelor testării, ignorînd, teoretic, resursele disponibile ale sistemului şi modul de utilizare în momentul execuţiei programului. Inconvenientul major al acestei tehnici este că sintaxa limbajului, compilatorul şi, eventual, monitorul trebuie, parţial, adaptate tipurilor de resurse ale unui sistem. Uneori este dificil de a specifica un tip de semnal sau o măsurătoare, prin stabilirea stării instrumentului ce participă la acţiune.

In urma analizei celor două tehnici se propune realizarea unui compromis, din care să rezulte un limbaj ce să permită elaborarea programelor bazîndu-se pe un control orientat spre elementul sub test, dar să asigure conservarea imediată şi completă a controlului resurselor sistemului. Soluţia este un mod de lucru în multinivel şi oferă următoarele avantaje :

- controlul direct al ansamblului de resurse;
- permite utilizarea programării structurate;
- capacitate mare de extensie.

Un limbaj ce se încadrează în prima tehnică este LITEST. Acesta este un limbaj mnemonic, de complexitate redusă, destinat testării funcţional-dinamice a plachetelor echipate cu circuite logice. Un vector de testare conţine descrierea stimulilor aplicaţi pe pinii de intrare, precum şi valoarea răspunsului obţinut, la un moment dat, pe pinii de ieşire. Configuraţia tuturor vectorilor se descrie cu mnemonice LITEST, de exemplu :

- IN Listă - specifică declararea pinilor ca pini de intrare;
- IH Listă, IL Listă - specifică atribuirea de nivele logice pinilor de intrare (IH - "1", IL - "0") etc.

Limbajul nu permite definirea şi utilizarea de variabile, simboluri, şiruri de caractere, ci numai de subprograme. Utilizarea de algoritmi de testare eficienţi este greoaie şi nu atinge parametrii optimi.

Datorită simplităţii limbajului, interpretorul asociat este rapid şi mai puţin complex.

Limbajul ATIAS [*4],[*5],[*20],[*35] , [* 46] este un limbaj de testare evoluat, permiţînd descrierea cerinţelor de testare ale elementului sub test, independent de caracteristicile sistemului de testare.

In diferitele aplicaţii, se utilizează următoarele nivele ale limbajului ATIAS :

- standardul ATLAS complet, ca şi configuraţie, sintaxă, semantică şi vocabular;
- un subset ATIAS - cuprinde o parte din standardul iniţial.

Elementele preluate sînt perfect identice cu cele din standardul original.

- limbaj ATLAS adaptat - plecînd de la standardul original s-au adaptat o serie de limbaje, în funcţie de caracteristicile aplicaţiei, posibilităţile sistemului de testare (tip calculator, memorie, viteză) etc.

Limbajul ATLAS permite definirea şi utilizarea de variabile, simboluri, şiruri de caractere, subprograme şi deci, implicit, a programării structurate.

Pentru ca limbajul să asigure aceleaşi facilităţi în toate fazele de existenţă ale unui program şi anume : de elaborare, punere la punct şi exploatare, este indicat ca, com-

pilatorul să fie de tip incremental. In acest caz, orice modificare în programul de testare solicită recompilarea numai a modulului afectat, dar în contextul stabilit pentru întregul program.

Caracteristicile limbajului ATLAS (în special sintaxa), la orice nivel, solicită construcţia unui compilator complex. Din acest motiv şi pentru a asigura o portabilitate adecvată, în cazul unui număr cît mai mare de sisteme de testare, este indicată efectuarea compilării în două faze. In prima fază, codul sursă ATLAS este translatat într-un cod intermediar, care nu depinde de sistemul pe care se abordează. A doua fază de compilare realizează translatarea, ţinînd cont de toate capabilităţile şi particularităţile sistemului de testare, pe care va fi exploatat programul. In felul acesta se asigură şi o modularitate şi un control efectiv al configurării sistemului.

Alocarea resurselor [*4],[*5],[*36],[*46] în faza de compilare realizează o optimizare a codului de execuţie, mai ales în ceea ce priveşte memoria solicitată şi timpul de exploatare. Aceasta se realizează printr-un proces bine delimitat în timp şi spaţiu şi constă în :

- selecţia şi descrierea resurselor;
- descrierea şi analiza semnalelor;
- alocarea propriu-zisă.

Ca dezavantaj, alocarea în faza de compilare este neeconomică, deoarece se alocă resurse şi pe ramuri ale programului, care, pentru un caz dat, nu sînt exploatate.

Alocarea în faza de execuţie este mult mai economică, dar duce la scăderea vitezei de execuţie.

Se apreciază că, compilarea în două faze oferă o soluţie de compromis, care încearcă să ţină cont de toate problemele legate de alocarea resurselor.

Pentru exploatare este indicată utilizarea unui procesor interpretativ, care oferă o serie de posibilităţi legate de modificarea codului de execuţie, element de importanţă majoră, mai ales în faza de localizare a defectelor elementului sub test.

Un limbaj ATLAS destinat testării hibride cuprinde următoarele tipuri de instrucţiuni :

- instrucţiuni de identificare şi control a programelor;
- instrucţiuni de declaraţie, prin care se realizează : atribuirea de nume simbolice pinilor, desemnarea pinilor numerici în diferite logici, fixarea formatului de afişare etc.;

- instrucţiuni de program propriu-zise, care pot fi :
  - instrucţiuni orientate test (CONECT, READ, SETUP etc.);
  - instrucţiuni procedurale (COMPARE, CAICULATE etc.);
  - instrucţiuni de control (GOTO, EXECUTE etc.);
  - definiţii şi apeluri de subprograme (CALL etc.);
  - instrucţiuni cu efect de temporizare (DELAY, PAUSE etc.).

In concluzie, se poate afirma că limbajele orientate spre testare oferă avantaje substanţiale în faza de elaborare şi exploatare a programelor de testare.

Pentru elaborarea automată a programelor de testare se utilizează generatoare adecvate, simulatoare, biblioteca de circuite şi biblioteca programelor de construcţie a vectorilor de testare. Schema cu care este echipată placheta, descrisă cu ajutorul unui limbaj simbolic, este simulată în memoria echipamentului de calcul. In felul acesta se asigură posibilitatea atribuirii de valori logice corecte variabilelor ce descriu comportarea ieşirilor schemei, cînd pe intrări se aplică vectorii de testare. Programul care realizează aceste atribuiri poartă numele de autoprogramator. Un ajutor substanţial prezintă simulatorul, în faza de localizare a defectelor, oferind în orice moment o hartă a stării tuturor nodurilor.

3.3. Elaborarea unui limbaj destinat descrierii schemelor logice [K1], [P2], [P4], [W2], [*4], [*5], [*6], [*8], [* 17], [*24], [*25].

Limbajul ce se propune permite descrierea componentelor pentru orice nivel de simulare, utilizînd o codificare simplă şi flexibilă, apropiată de limbajul utilizatorilor de circuite logice.

Schema logică de pe plachetă poate fi descrisă funcţional, specificîndu-i intrările, ieşirile, componentele integrate sau/şi discrete din configuraţia sa şi interconexiunile dintre acestea. Parametrii statici, dinamici, funcţionali, precum şi tipul pinilor fiecărei componente se păstrează în bibliotecă. In felul acesta se asigură o analiză strictă a modului de utilizare a componentelor.

Informaţiile sursă oferite de utilizator se grupează pe secţiuni, după cum urmează :

- secţiunea de comentariu - este deschisă de caracterul %C şi poate conţine orice caracter alfanumeric. Această secţiune nu

este prelucrată şi poate fi introdusă în orice punct al programului;

- secţiunea interconexiunilor între componente - este deschisă de caracterul $D. O linie din program conţine un şir de identificatori, separaţi prin virgulă, sfîrşitul liniei fiind marcat printr-un caracter special. Lungimea liniei este nelimitată. Ca identificatori pot apare :

  -[I][$n_1n_2n_3$] - specifică o intrare pe plachetă, prin conector; $n_1n_2n_3$ - este numărul pinului utilizat ca intrare, în exprimare zecimală. Acest număr poate fi format din maximum trei cifre zecimale, sfîrşitul identificatorului fiind marcat prin virgulă sau un caracter special. Exemple : I2, I49;

  -[Ø][$n_1n_2n_3$] - specifică o ieşire de pe plachetă prin conector; $n_1n_2n_3$ - are aceeaşi semnificaţie şi configuraţie ca şi în cazul pinului de intrare. Exemple : Ø6, Ø92;

  - [#][Nume compon.][P][n_1n_2] - specifică componenta şi numărul pinului ce intră în conexiune; identificatorul "Nume compon." specifică o componentă integrată şi este format din litera Y urmată de maximum trei cifre zecimale; n_1n_2 - precizează, în exprimare zecimală, numărul pinului componentei şi este format din maximum două cifre zecimale, sfîrşitul exprimării fiind delimitat de virgulă sau un caracter special; caracterul special "#" arată că pinul căruia îi este ataşat face parte dintr-o legătură de reacţie. Această precizare, de excepţie, se solicită pentru a permite atribuirea de valori iniţiale intrărilor care fac parte din legăturile de reacţie. Dacă pinul componentei nu este utilizat în acest scop, caracterul special "#" lipseşte. Exemple : Y4P6, Y8P4, Y8P5;

  - [TE / TB / TC] [$n_1n_2n_3$] - specifică conexiunile pentru tranzistoare. Se precizează locul legăturii (în emitor - TE, în colector - TC sau în bază - TB) şi numărul tranzistorului din legătură. Acest număr este tot în exprimare zecimală şi are maximum trei cifre, după ultima, urmînd virgula sau un caracter special. Exemple : Y2P3, TB93, TB42;

- $\begin{bmatrix} RI \\ R\emptyset \end{bmatrix} [n_1 n_2 n_3]$ - specifică conexiunile pentru rezistențe. Pentru verificare și exploatare, în faza de simulare se consideră conexiunea RI, ca intrare, iar conexiunea $R\emptyset$, ca ieșire; $n_1 n_2 n_3$ - reprezintă numărul de ordine al rezistenței, în exprimare zecimală. Această valoare poate fi formată din maximum trei cifre, după ultima, urmînd virgula sau un caracter special. Exemplu : Y4P9, RI42;
- $\begin{bmatrix} KI \\ K\emptyset \end{bmatrix} [n_1 n_2 n_3]$ - specifică conexiunile pentru condensatoare. Pentru aceleași raționamente ca și în cazul rezistențelor, KI specifică conexiunea de intrare iar $K\emptyset$, conexiunea de ieșire; $n_1 n_2 n_3$ - reprezintă numărul de ordine al condensatorului, exprimat zecimal și format din maximum trei cifre, după ultima, urmînd virgula sau un caracter special. Exemple : Y3P2, Y4P5, KI100;
- $\begin{bmatrix} VCC \\ GND \end{bmatrix}$ - precizează legături fixe la alimentare:
- [Nume semnal] - desemnează, opțional, un nume semnalului ce circulă prin interconexiunea la care se referă; este format din maximum șase caractere litere sau cifre, primul caracter fiind literă; se introduce după ultimul identificator din linie și este precedat de un caracter special. Exemple : Y2P3, Y4P5, Y6P8, @ CLOCK.

Intr-o linie sînt cuprinse toate conexiunile aceluiași nod.

- secțiunea de specificare a tipului componentelor - este deschisă de caracterul *E. O linie din această secțiune conține un șir de identificatori, care reprezintă numele componentelor, despărțiți prin virgulă. In continuare urmează semnul egal și specificarea tipului componentei. Tipul componentei se descrie astfel :
  - maximum trei litere, ca primă literă fiind excluse literele R și K, urmate de maximum cinci cifre zecimale - pentru componentele integrate;
  - litera R urmată de maximum patru cifre zecimale, care specifică valoarea în KOhmi - pentru rezistențe;
  - litera K urmată de maximum patru cifre zecimale, care specifică valoarea în nF - pentru condensatoare:

  Exemple : Y2,Y4,Y6,Y7 = CDP4?C;
  R2,R8,R9 = R2?;
  K5,K7,K4 = KI?C.

Sfîrşitul descrierii circuitului se specifică prin caracterul $F.

Cu ajutorul limbajului de mai sus se prezintă, în fig.3.2 (b), un exemplu de descriere a schemei din fig.3.2(a).

Sfîrşitul unei secţiuni este marcat de începutul altei secţiuni sau de caracterul $F.

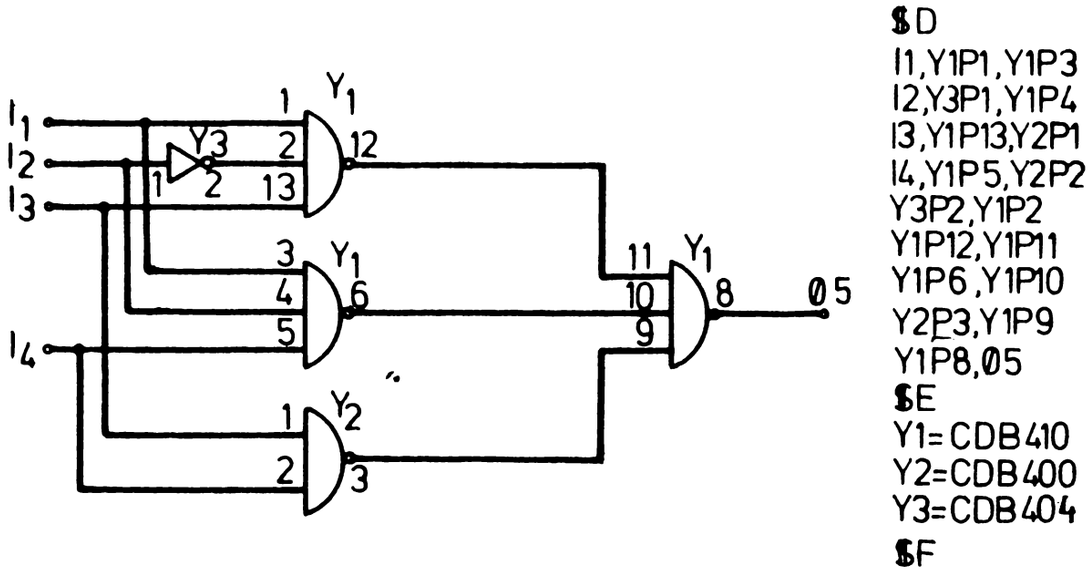


Fig.3.2.

După cum se observă, limbajul adoptat este practic, orientat spre circuit şi uşor de abordat de cei care lucrează în domeniu. Sintaxa s-a căutat să fie cît mai simplă, pentru a reduce timpul de lucru al analizorului sintactic.

3.4. Analizorul sintactic al limbajului

In faza de analiză sintactică a textului sursă se verifică specificarea corectă a definirilor, ca lungime şi conţinut, dublele definiri, formatul secţiunilor şi a liniilor din secţiuni. Se indică locul erorii în fiecare linie, iar la sfîrşit se afişează numărul total de erori. Rezultatul prelucrării îl constituie generarea a două tabele şi anume : TAB1 - tabelul inter-

conexiunilor şi TAB2 - tabelul de specificare al tipului componentelor. Aceste tabele se generează, în paralel cu analiza sintactică. Liniile din textul sursă, care prezintă cel puţin o eroare sintactică, nu sînt incluse în tabele. Acelaşi lucru se întîmplă cu separatorii, caracterele speciale, blancurile şi comentariile. In concluzie, se reţin numai informaţiile strict necesare specificării exacte şi corecte a interconexiunilor şi tipurilor componentelor. In acest fel se urmăreşte o utilizare cît mai raţională a capacităţii de memorare a echipamentului de calcul şi creşterea vitezei de prelucrare în fazele următoare.

In fig.3.3 se prezintă cîte o linie din fiecare tabel.

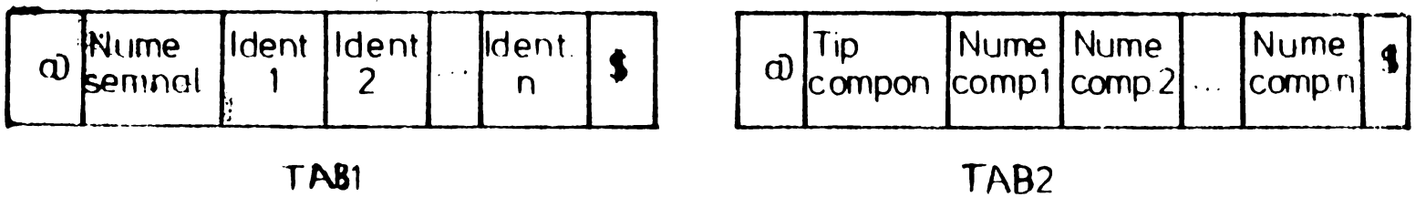


Fig.3.3.

Fiecărui nod descris în program îi corespunde o linie în TAB1 şi fiecărui tip de componentă îi corespunde o linie în TAB2. Inceputul unei linii este specificat prin caracterul special @ iar sfîrşitul tabelului respectiv, prin caracterul $.

Existenţa unor erori nu presupune, automat, oprirea prelucrării, odată cu încheierea acestei faze. Utilizatorul va fi cel care va decide oprirea sau continuarea cu faza de execuţie, în funcţie de locul, gravitatea şi numărul erorilor, deoarece, în mod cert, există situaţii cînd o eroare sintactică afectează în măsură mai mică comportarea, în ansamblu, a schemei sau afectează o parte bine delimitată din configuraţia sa, eventual, care se poate izola.

In fig.3.4 se prezintă schema bloc a modulului de construcţie a tabelelor TAB1 şi TAB2. Notaţiile utilizate au următoarele semnificaţii :

$DESC_i$ - tabloul care conţine caracterul în curs de analiză;
ALFAB - variabilă care conţine caracterul ce se analizează, dacă aceasta este literă;
CIFRA - variabilă care conţine caracterul ce se analizează, dacă aceasta este cifră;
BLANC - variabilă care conţine caracterul ce se analizează, dacă aceasta este blanc (b);

CS - variabilă care conţine caracterul ce se analizează, dacă acesta este un caracter special;

ETER - rutină pentru tratarea erorilor.

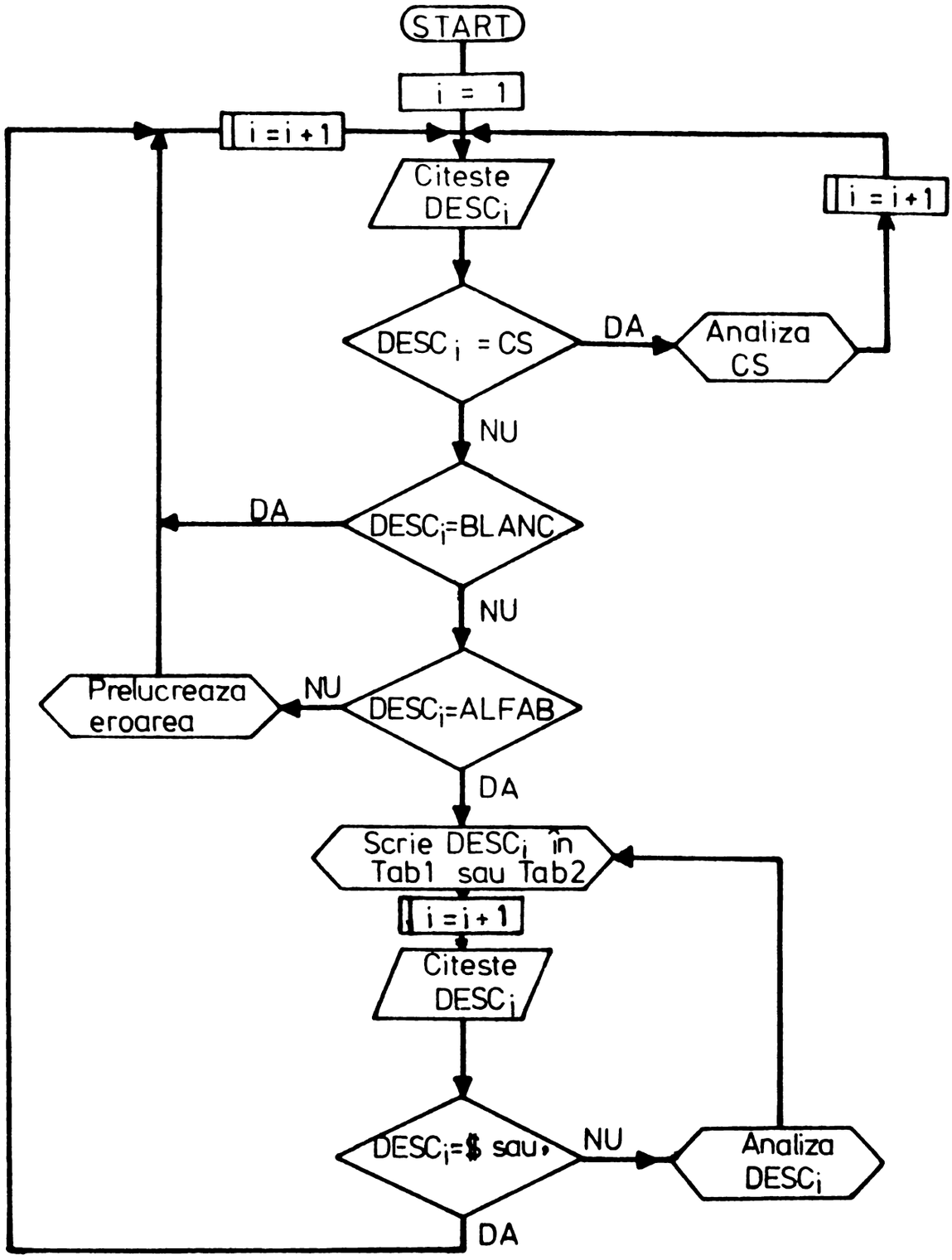


Fig.3.4.

3.5. Biblioteca de circuite

Biblioteca de circuite asigură o descriere funcţională şi parametrică a circuitelor integrate logice, a celor mai utilizate tranzistoare comerciale şi a unor circuite realizate cu componente discrete.

Parametrii caracteristici ce-i furnizează biblioteca, pentru circuitele integrate logice, sînt : numărul de intrări, numărul de ieşiri, factorii de încărcare la intrare şi la ieşire, tipul etajului de ieşire (în contratimp, cu colector deschis sau cu trei stări), timpul mediu de propagare, tipul fiecărui pin.

Descrierea funcţională se realizează cu ajutorul unor rutine aferente fiecărui circuit integrat. Alte rutine sînt destinate calculului circuitului inversor realizat cu componente discrete, a răspunsului circuitelor RC trece-sus şi a celor trece-jos.

Modelele conţinute în bibliotecă se pot diviza în :

- modele de bază - caracterizează circuitele integrate SSI, cum sînt porţile de orice tip, circuitul inversor, circuitul amplificator etc., precum şi circuitele combinaţionale MSI, ca : decodificatoare, multiplexoare etc.;
- modelele de memorie - în această categorie intră bistabilele, registrele, numărătoarele şi memoriile;
- modele speciale - cum ar fi conectorul;
- modelele de timp - realizează diferite întîrzieri programate.

Pentru fiecare model, construit pe baza tabelelor de adevăr, prezentate în cap.2, pinii sînt partiţionaţi în intrări de date şi de comandă şi ieşiri de date, respectiv de control. Mai ales în cazul circuitelor secvenţiale, se fac precizări amănunţite asupra fiecărei intrări de comandă şi a momentului de basculare : pe frontul anterior, posterior sau pe palierul semnalului de tact.

Modelele de memorie, în afară de bistabile, au o lungime variabilă, în funcţie de tipul memoriei ce se prelucrează. In felul acesta se obţine o importantă economie de spaţiu.

Diferitele modele de întîrzieri permit o serie de facilităţi în faza de simulare, legate de analiza comportării dinamice a circuitelor.

Modelul conectorului realizează transferul semnalelor între schema de pe plachetă şi sistemul de testare, fiind descris

ca un amplificator cu factorul de amplificare egal cu unitatea.

După cum se observă, în descrierea conţinută în bibliotecă, nu se insistă asupra configuraţiei electronice a circuitelor integrate. In felul acesta, economia de memorie solicitată şi creşterea vitezei, în faza de execuţie, reprezintă cele mai importante cîştiguri.

In urma apelurilor la bibliotecă, pentru descrierea schemei de pe plachetă, se poate face o analiză exactă, pe lîngă cea sintactică, a corectitudinii utilizării şi din punctul de vedere al parametrilor caracteristici. Această analiză este realizată în timpul procesului de simulare.

Tuturor intrărilor circuitelor integrate din bibliotecă li se atribuie, iniţial, o valoare logică caracteristică stării de repaus, valoare care să nu influenţeze starea ieşirii, în situaţia în care intrarea respectivă rămîne nespecificată în timpul procesului de simulare.

Parametrii circuitelor din bibliotecă caracterizează familia TTL şi respectă datele de catalog.

Biblioteca de circuite este păstrată pe un suport periferic, de unde este apelată la nevoie.

In fig.3.5 se prezintă modul de organizare al bibliotecii : a) repertorul bibliotecii; b) descrierea circuitului integrat.

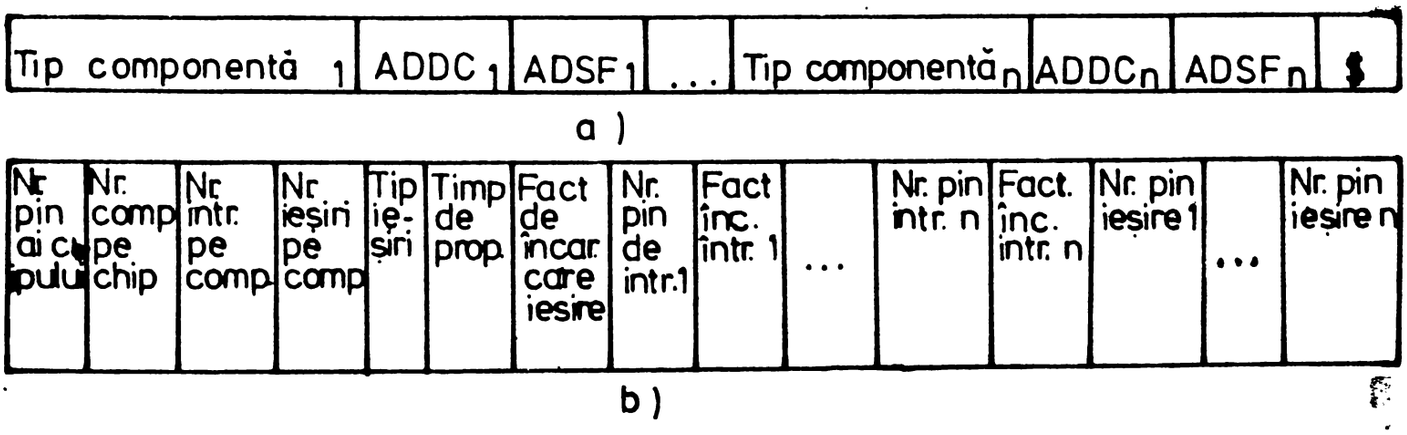


Fig.3.5.

Repertorul bibliotecii conţine toate tipurile de componente din bibliotecă, alături de adresele la care se află descrierea circuitului integrat aferent (ADDC) şi rutina ce caracterizează funcţionarea logică a componentei (ADSF). Incheierea reper-

torului se indică cu caracterul special $. Un exemplu de descriere a circuitului integrat CDB4oo se prezintă în fig.3.6. Indicaţiile sînt exprimate în zecimal, iar timpul de propagare în ns. Semnificaţia specificaţiilor din rubrica "Tip ieşire" este : N - ieşire normală, Ø - ieşire cu colector deschis, S - ieşire cu trei stări. Factorii de încărcare sînt exprimaţi în unităţi de încărcare, o unitate reprezentînd sarcina impusă de o intrare a unei porţi logice TTL.

Nr pini ai chip	Nr comp ale chip	Nr intr pe comp	Nr ieşin pe comp	Tip ie-sin	Timp de prop	Fact de inc ieşire	Nr pin intr 1	Fact inc intr 1	Nr pin intr 2	Fact inc intr 2	Nr pin ies	Nr pin intr 1	Fact inc intr 1	Nr pin intr 2	Fact inc intr 2	Nr pin ies		Nr pin intr 1	Fact în-căr	Nr pin intr	Fact în-căr. intr2	Nr pin ies
14	4	2	1	N	10	10	1	1	2	1	3	4	1	5	1	6		10	1	9	1	8

Fig.3.6.

Adăugarea de noi circuite integrate sau modificarea celor existente deja în bibliotecă nu implică nici o problemă deosebită.

3.6. Procesorul de defecte

Bazîndu-se pe descrierea schemei cu care este echipată placheta, procesorul de defecte realizează o hartă a tuturor defectelor de tipul forţărilor, ce pot apare în schema realizată pe plachetă. Pentru reducerea numărului de defecte, foarte importantă este etapa de stabilire a claselor de defecte echivalente, clase stabilite plecînd de la configuraţia schemei.

O primă clasă de echivalenţă este aferentă tipului de componentă utilizată. Pentru o componentă integrată, la care semnalul de ieşire nu este inversat faţă de cel de la intrare, defectele de acelaşi tip de la intrare şi ieşire sînt echivalente. Dacă semnalul este inversat, un defect, la intrare, de tipul f0 este echivalent cu un defect, la ieşire, de tipul f1 şi invers. Se specifică faptul că este vorba de defectele de pe căile de propagare, indiferent că acestea sînt intrări de date sau de comandă, respectiv ieşiri de date sau de control. Definiţia se referă la tipurile de MPA prezentate în cap.2.

O altă clasă de echivalenţă se poate stabili din analiza configuraţiei schemei de pe plachetă.

Se consideră două mulţimi α şi β formate din componente integrate şi definite astfel :

- mulţimea α conţine n componente, iar mulţimea β, m componente astfel :

$$\alpha_i \equiv \alpha_j \ , \quad \text{pentru } i,j = 1,n$$
$$\beta_i \equiv \beta_j \ , \quad \text{pentru } i,j = 1,m \qquad (3.1)$$
$$\alpha_i \not\equiv \beta_j \ , \quad \text{pentru } i = 1,n;\ j = 1,m.$$

Defectele de acelaşi tip, care apar la ieşirile de acelaşi tip (de date, respectiv de control) ale componentelor mulţimii α sînt echivalente, dacă respectivele ieşiri atacă împreună, intrările de acelaşi tip ale fiecărei componente a mulţimii β. In fig.3.7 defectele de acelaşi tip de la ieşirile porţilor Y2 şi Y3 sînt echivalente.

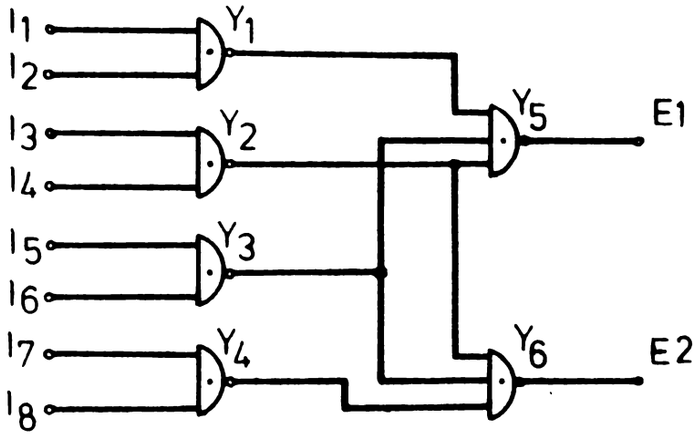


Fig.3.7.

In funcţie de configuraţia schemei se pot stabili şi alte clase de echivalenţă.

Se apreciază că utilizînd numai cele două definiri de mai sus, numărul total de defecte considerate se reduce cu mai mult de 50%.

Pentru construirea vectorului de testare aferent, se reţine un singur defect din fiecare clasă de echivalenţă.

O linie din biblioteca de defecte, creată de procesorul de defecte, se prezintă în fig.3.8.

Pentru fiecare defect, simulatorul va crea un model ce va fi utilizat în faza de stabilire a vectorului (vectorilor) de testare. Un defect prelucrat este eliminat din biblioteca de defecte. Pentru fiecare vector de testare se prezintă clasa de defecte echivalente asociate.

Caracter special	Identificator pin	Tip defect	Defecte echivalente				
			Ident. pin	Tip defect	...	Ident. pin	Tip defect
@	Y2P3	F0	Y3P4	F1		Y4P6	F0

Fig.3.8.

3.7. Biblioteca programelor de construire a vectorilor de testare

In general, această bibliotecă conţine, programaţi, diferiţi algoritmi ce vor fi utilizaţi pentru construirea automată a vectorilor de testare, cum ar fi : algoritmul D, algoritmul lui Poage, algoritmul lui Armstrong etc.

In cazul de faţă, biblioteca conţine, programat, algoritmul prezentat în cap.2.

Tot în această bibliotecă se află programate diferite rutine de utilitate generală, necesare în faza de construire a vectorilor de testare, avînd drept scop **facilitarea** acestui proces. Acestea sînt :

- rutina TACT - generează o serie repetitivă de impulsuri, a căror perioadă, durată şi număr sînt programabile;
- rutina NIV - generează semnale de tip nivel, cu valoarea egală cu "1", "0" sau "X", pentru durate de timp programate;
- rutina ALEATOR - generează secvenţe de semnale pseudoaleatoare;
- rutinele INCR şi DCR realizează o numărare în sens crescător, respectiv descrescător, între o valoare iniţială şi una finală, cu un anumit pas, elemente care se specifică prin program.

In cazul plachetelor echipate şi cu circuite integrate LSI se utilizează acelaşi algoritm de generare, cu deosebire că în cazul celor programabile (microprocesoare) se execută mai întîi programul de ordonare a succesiunii de activare a instrucţiunilor sau comenzilor/modurilor de lucru, după care, pe baza rezultatelor obţinute, se abordează faza de construire a vectorilor de testare.

Dintre algoritmii mai cunoscuţi, în bibliotecă se află programat cel al lui Poage. S-a adoptat aceasta, deoarece permite

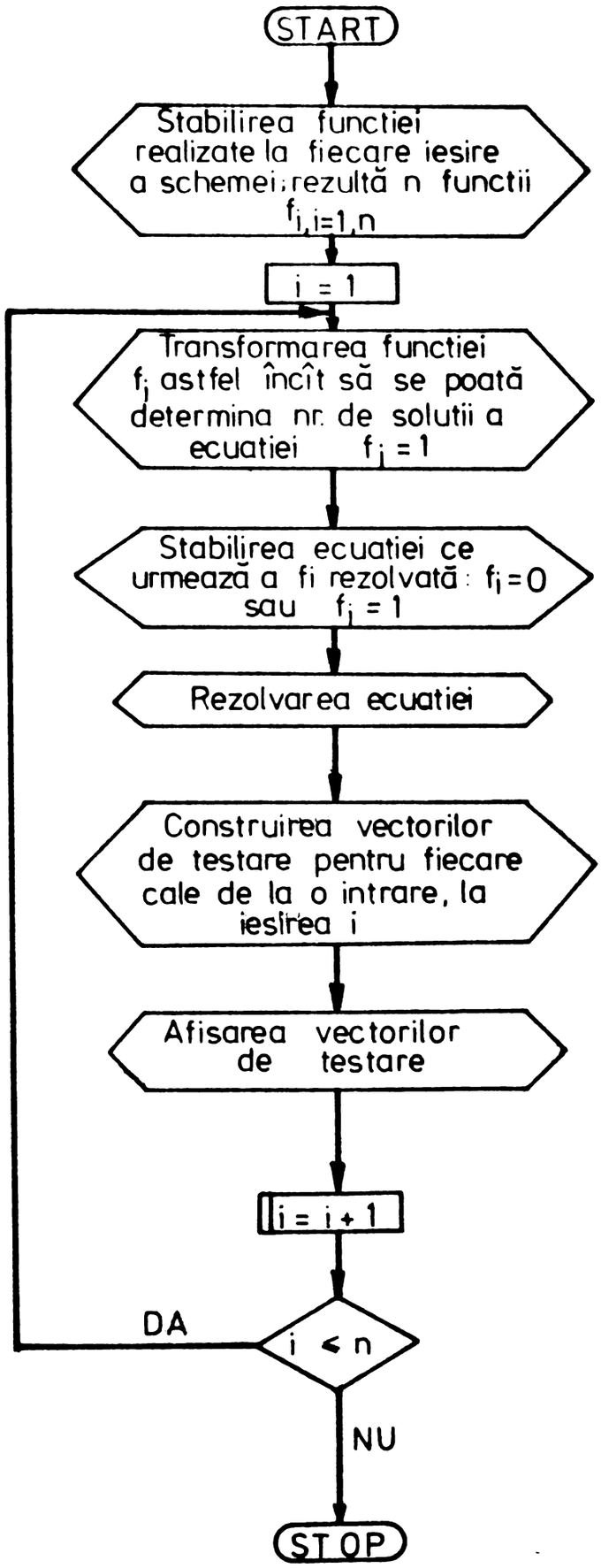

START
Stabilirea functiei realizate la fiecare iesire a schemei; rezultă n functii $f_i, i=1,n$
i = 1
Transformarea functiei f_i astfel încît să se poată determina nr. de solutii a ecuatiei $f_i = 1$
Stabilirea ecuatiei ce urmează a fi rezolvată: $f_i = 0$ sau $f_i = 1$
Rezolvarea ecuatiei
Construirea vectorilor de testare pentru fiecare cale de la o intrare, la iesirea i
Afisarea vectorilor de testare
i = i + 1
DA
i < n
NU
STOP

Fig.3.9.

construirea întregului set al vectorilor de testare pentru o schemă dată şi poate fi utilizat, cu puţine restricţii şi adaptări, în testarea unei game variate de tipuri de scheme cu care sînt echipate plachetele.

Modul de lucru, cînd se utilizează algoritmul din cap.2, pentru construirea vectorilor de testare, se prezintă în schema bloc din fig.3.9.

Cînd se apelează algoritmul lui Poage,pentru stabilirea vectorilor de testare,se utilizează acelaşi mod de introducere a funcţiilor ce descriu comportarea ieşirilor schemei realizată pe plachetă. De asemenea şi pentru afişarea vectorilor se utilizează aceeaşi rutină.

3.8.**Procesorul** de simulare [C2],[P4],[T3],[U1],[U2], [V2],[W1],[W2],[*16].

Prin simulare numerică, utilizînd modelul sistemului fizic, se realizează analiza comportării în timp a acestuia, cînd este atacat, la intrare, cu un set de stimuli.

In cazul nostru, sistemul fizic este reprezentat de schema logică cu care este echipată placheta, iar prin simulare se urmăreşte realizarea unei testări adecvate a plachetelor echipate, utilizînd o referinţă virtuală.

Totodată, simulatorul constituie un instrument important în elaborarea sau îmbunătăţirea programelor de testare, în localizarea defectelor pe plachetele echipate, iar în faza de proiectare poate fi utilizat pentru analiza performanţelor funcţionale şi evaluarea proiectării din punctul de vedere al testabilităţii.

Modelele virtuale, realizate prin soft, reţin cele mai importante caracteristici, în special funcţionale, ale circuitelor. Crearea şi analiza modelului nu presupune, neapărat, cunoaşterea modului cum răspunde circuitul fizic la diferiţi stimuli, iar nivelul lui de complexitate poate fi dezvoltat gradat, în funcţie de necesităţile şi posibilităţile evaluării.

Performanţele obţinute au impus această tehnică, faţă de simularea prin hard, cînd se urmăreşte utilizarea, ca referinţă, a unei plachete presupusă corectă. Realizarea unei astfel de referinţe ridică probleme tehnice şi economice deosebite, mai ales cînd se lucrează cu componente de mare complexitate, cum sînt cele LSI/VLSI.

Pentru pregătirea operaţiei de testare, existenţa unui model soft al circuitului fizic permite injectarea de defecte în

orice nod, construirea vectorilor de testare, ce asigură detectarea şi localizarea defectelor şi evaluarea eficacităţii acestora.

In faza de testare propriu-zisă, un astfel de model constituie un ajutor preţios pentru localizarea rapidă a defectelor la nivelul unui nod.

In elaborarea simulatoarelor se utilizează două tehnici de bază :

- simularea pe bază de cod executabil - în acest caz, toate ecuaţiile ce descriu funcţionarea, ordonate astfel încît să simuleze propagarea semnalelor de la intrare la ieşire, fără reacţii, vor fi executate pentru fiecare set de stimuli, aplicat la intrări. Orice buclă se va întrerupe. Acest mod de lucru nu permite realizarea unei analize precise din punct de vedere al timpului de propagare, iar durata simulării este relativ mare. Totodată, o modificare în descriere cere prelucrarea întregului program. Se asigură simularea numai a două valori logice şi anume "0" şi "1";
- simularea pe bază de tabele - această tehnică, în care informaţiile sînt organizate sub formă de tabele, în funcţie de tipul fiecăreia, asigură o eficienţă sporită, generalitate, extensibilitate şi modularitate în exploatare. Posibilitatea utilizării a diverse modele de întîrzieri oferă facilităţi importante pentru testarea în regim dinamic. Se pot simula trei sau patru stări logice : "1", "0", "X" - starea incertă, "Z" - starea de înaltă impedanţă. Acestea asigură capabilităţi noi legate de : detectarea hazardului, iniţializarea circuitelor, simularea în situaţia în care starea iniţială a unor componente nu se cunoaşte exact, descrierea modului de lucru a schemelor formate din componente cu ieşiri legate împreună etc.

Simulatorul ce se prezintă are o configuraţie tabelară şi utilizează, în exploatare, modelul circuitului realizat pe baza MPA prezentate în cap.2.

Pentru fiecare identificator din tabelul TAB1, al cărui tip este precizat în tabelul TAB2, se face un apel la biblioteca de circuite, de unde se extrag informaţiile : tipul pinului (de intrare sau de ieşire), tipul ieşirii (în contratimp, cu colector deschis, cu trei stări), factorii de încărcare, timpul de propagare. Cu aceste elemente se construiesc : tabelul ieşirilor, TAB3, tabelul destinaţiilor, TAB4 şi tabelul intrărilor, TAB5. Configuraţia acestora se prezintă în fig.3.1o. Caracterul speci-

al # se introduce numai dacă respectiva destinaţie constituie o legătură de reacţie.

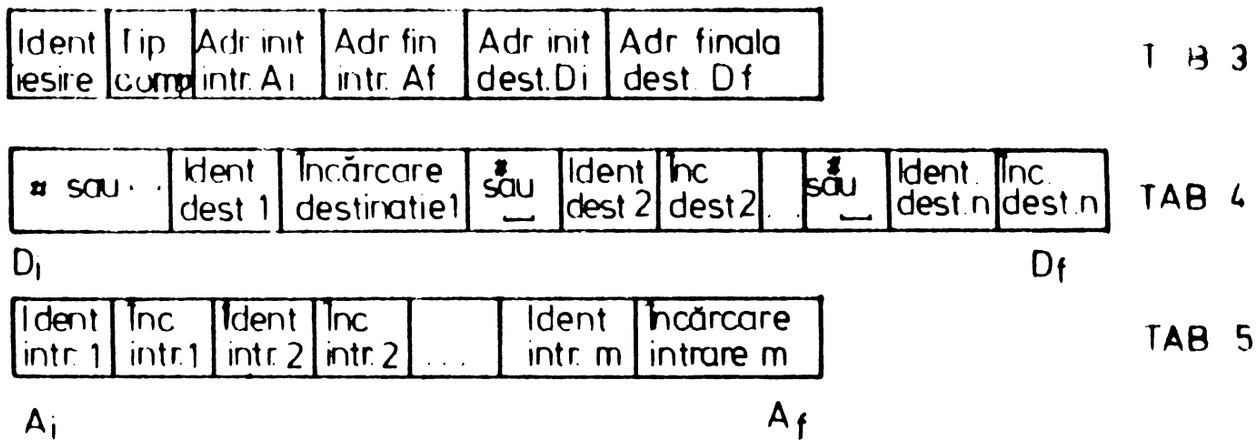


Fig.3.1o.

Utilizarea adreselor în locul indicatorilor propriu-ziş asigură o manevrabilitate sporită asupra cîmpurilor respective, mai ales în cazul în care se fac diferite modificări şi actualizări.

In continuare, pe baza tabelului TAB3, se construieşte tabelul TAB6, identic ca şi configuraţie cu aceasta, dar avînd ieşirile ordonate în sensul de propagare al semnalelor. Ordonarea s-a făcut prin împărţirea componentelor pe nivele, astfel :

- în primul nivel intră componentele ale căror intrări sînt legate numai la pinii conectorului;
- în al doilea nivel intră componentele ale căror intrări sînt legate la ieşirile componentelor din primul nivel şi, eventual, la pinii conectorului;
- în al treilea nivel intră componentele ale căror intrări sînt legate la ieşirile componentelor din nivelul anterior şi, eventual, din celelalte nivele inferioare sau la pinii conectorului şi aşa mai departe.

Existenţa unor legături de reacţie nu influenţează partiţionarea în nivele, acestora atribuindu-li-se valori iniţiale, bine precizate.

Tabelul TAB6 descrie perfect fiecare nod al circuitului.

In această fază se fac o serie de verificări legate de utilizarea corectă a fiecărei componente din configuraţia schemei cu care este echipată placheta. Orice identificator prezent în tabelul TAB1, dar neprecizat în tabelul TAB2 este considerat element nedefinit. Apariţia în TAB2 dar neutilizarea în TAB1 duce

la considerarea identificatorului respectiv, ca element nereferenţiat. Tot acum se verifică şi se semnalează ca eroare, dublele definiri, adică acelaşi nume de element atribuit la componente de tipuri diferite.

In urma apelului la bibliotecă se verifică : existenţa în bibliotecă, numărul de pini ai circuitului integrat, numărul maxim de intrări şi de ieşiri, încărcările, legarea corectă a ieşirilor. Se consideră că ieşirile legate împreună realizează funcţia logică SI. Acest tip de legare se admite numai pentru ieşiri de tipul cu colector deschis sau cu trei stări. Din acest motiv se semnalează ca şi eroare apariţia în aceeaşi linie a tabelului TAB1 a mai mult decît o ieşire, care nu fac parte din categoria celor specificate mai sus.

In continuare se construieşte TAB7 - tabelul cu numele simbolurilor. Acest tabel face corespondenţa între numele identificatorilor şi adresele la care se găsesc valorile acestora. Pentru crearea tabelului TAB7 se caută numele ieşirii în tabelul TAB6 şi i se rezervă atîtea locaţii de memorie într-un tabel ce conţine valorile simbolurilor, **TAB7** , cîţi pini are circuitul integrat a cărui ieşire s-a extras din TAB6. In tabelul TAB7 se introduce numele şi adresa primului pin. Modul de lucru se prezintă în schema bloc din fig.3.11.

Folosind tabelele TAB4, TAB5 şi TAB7 se construiesc alte două tabele TAB8 şi TAB9, care, de fapt, vor lua locul tabelelor TAB4 şi TAB5. Tabelele TAB8 şi TAB9 vor conţine în locul numelui, adresa reală la care se găsesc valorile variabilelor respective, adresă extrasă din VSIME. Modul de lucru şi configuraţia tabelelor se prezintă în fig.3.12.

Ultimul tabel creat este TAB1o, care corespunde cu tabelul TAB6 ca şi configuraţie, dar conţine direct adresele reale ale elementelor ce vor fi utilizate în faza de execuţie şi anume: adresa rutinei ce permite calculul funcţiei realizată de circuitul logic a cărui ieşire se evaluează, adresele valorilor semnalelor aplicate pe intrări etc. Modul de lucru şi configuraţia acestui tabel se prezintă în fig.3.13.

In faza de simulare propriu-zisă se apelează tabelul TAB1O pentru evaluarea fiecărei ieşiri.

Crearea gradată a tabelelor în faza de pregătire a operaţiei de simulare şi apoi utilizarea unui număr redus de tabele, în faza de simulare propriu-zisă, contribuie la asigurarea unei

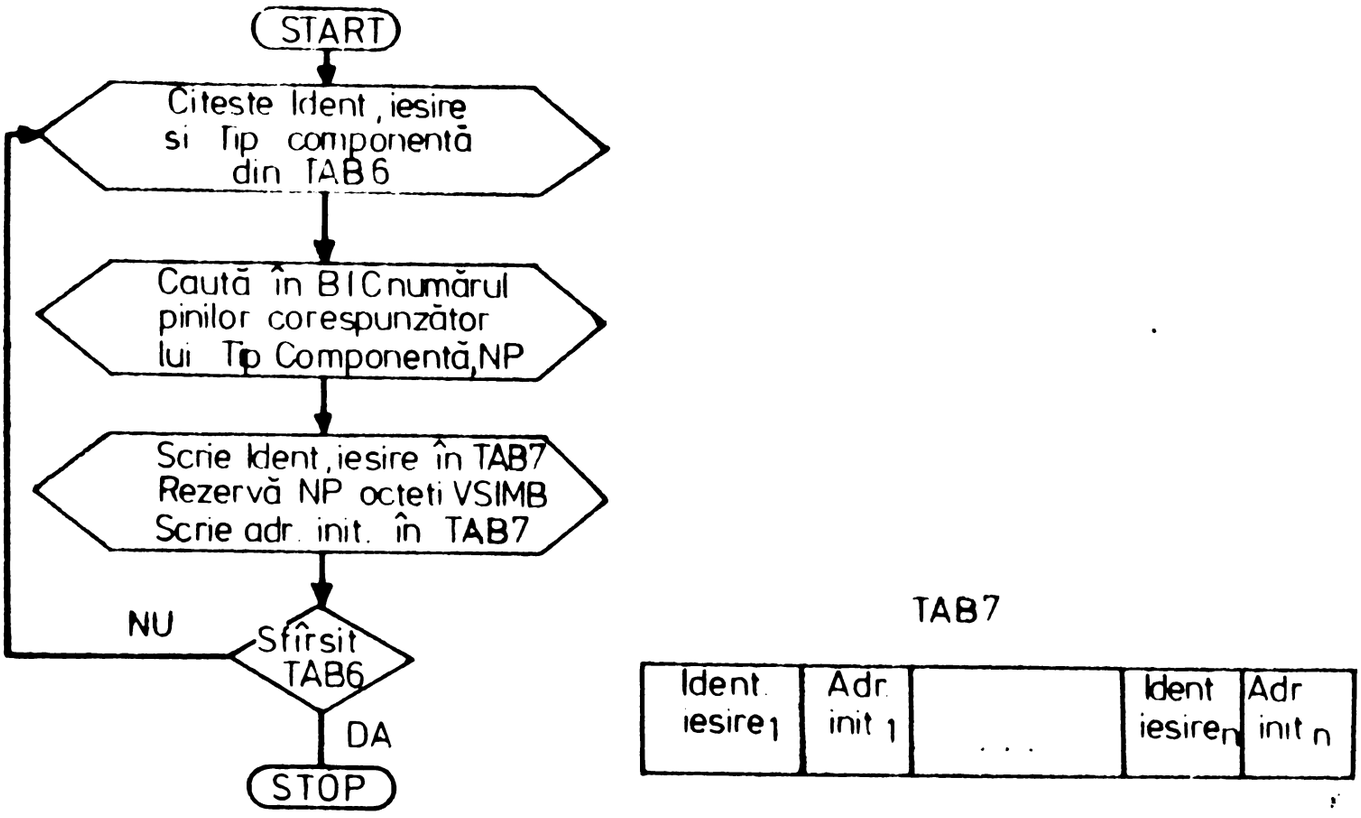


Fig.3.11.

manevrabilităţi ugoare a informaţiilor, pe de o parte şi la creşterea vitezei de caloul, pe de altă parte. Totodată apelul şi modificarea unor informaţii, deja depuse în tabele, se realizează foarte simplu şi rapid.

Se permite simularea a patru stări. Pentru codificare se utilizează doi biţi şi anume : 00-"0", 11-"1", 01-"X", 10-"Z".

"X" poate pune în evidenţă existenţa unei stări incerte, care apare la o tranziţie 0 → 1 sau 1 → 0. Regulile de operare, stabilite pentru această stare, sînt următoarele :

$$X + 0 = X \; ; \quad X.0 = 0 \; ; \quad \overline{\overline{X}} = X \; ; \qquad (3.2)$$
$$X + 1 = 1 \; ; \quad X.1 = X$$

Pentru a facilita modul de lucru, fiecare bit al unei stări se păstrează într-un registru. In acest caz, pentru efectuarea operaţiilor logice dintre diferitele stări, se apelează di-

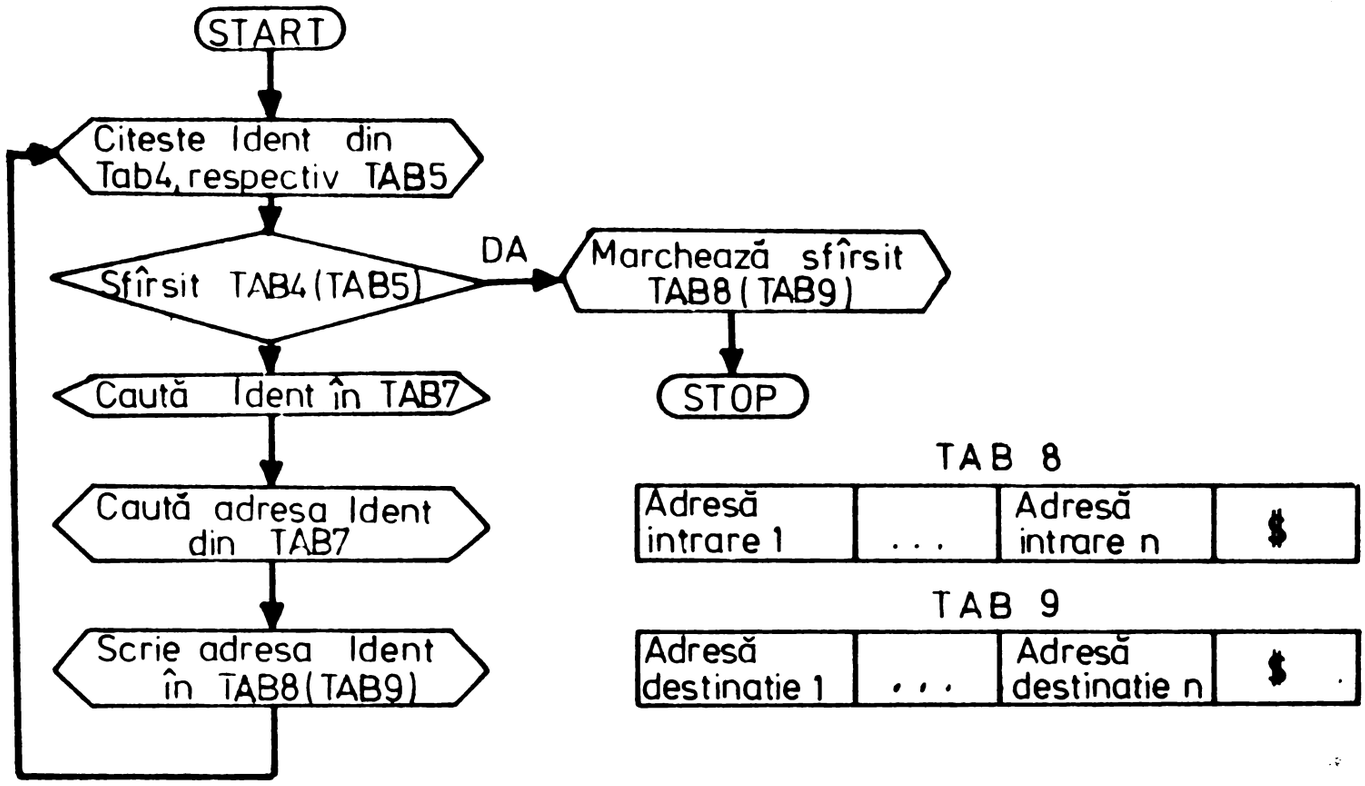
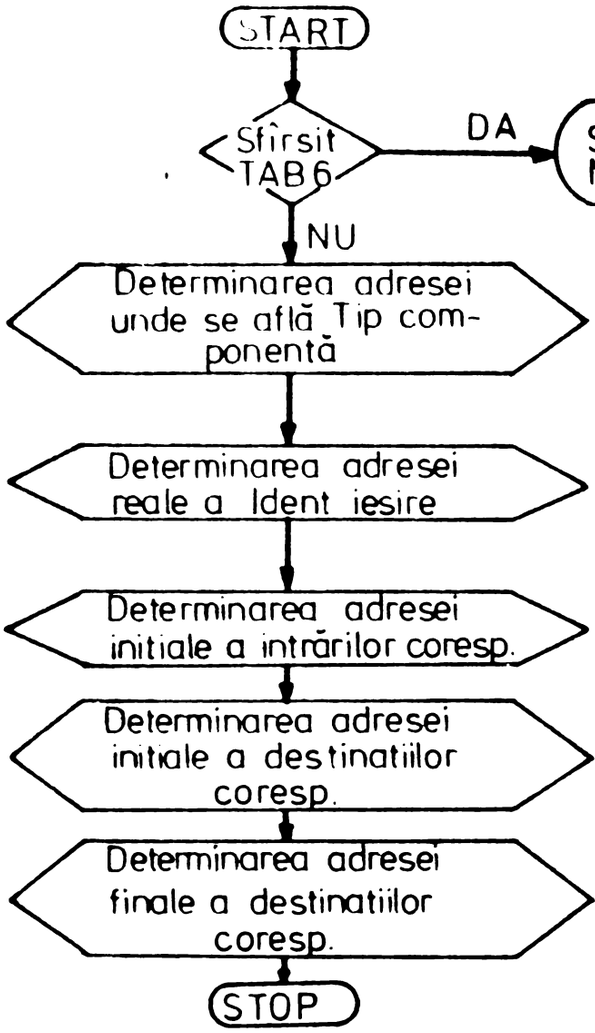


Fig.3.12.

rect la instrucţiunile de asamblare destinate acestui scop, care vor opera asupra registrelor luate două cîte două, după cum se prezintă în fig.3.14. Operanzii sînt A, B şi C. Operaţiile de complementare şi SAU-EXCLUSIV nu au corespondent direct în setul de instrucţiuni de asamblare. Pentru aceste două operaţii se utilizează rutine speciale de prelucrare. Deoarece inversarea unei stări necunoscute duce la o stare, de asemenea, necunoscută, în cazul apariţiei acestei operaţii, rutina asociată va realiza o dublă complementare a fiecărui registru, adică :

$$\text{- dacă } C = X, \quad \bar{C} \rightarrow \begin{matrix} R_{31} = \overline{\overline{R_{31}}} \\ R_{32} = R_{32} \end{matrix} \qquad (3.3)$$

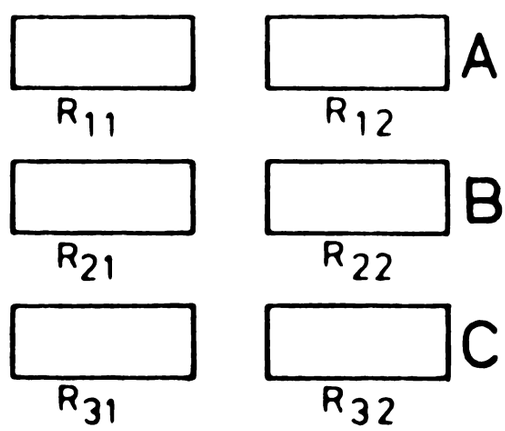
Starea "X" oferă o foarte bună reprezentare a stării de tranziţie a unui circuit, permiţînd detectarea hazardului logic, validarea iniţializării în situaţia în care starea iniţială nu



Tab 10

Salt la adresa pt. Tip compon.	Adr. init intr.	Adresa iesire	Adrese destinatii	
			Adresa initială	Adresa finală

Fig.3.13.



Dacă $A,B \in (0,1,x)$

$C = A + B \Rightarrow R_{31}=R_{11}+R_{21}$

$R_{32}=R_{12}+R_{22}$

$C = A \cdot B \Rightarrow R_{31}=R_{11}\cdot R_{21}$

$R_{32}=R_{12}\cdot R_{22}$

Dacă $A,B \in (0,1)$

$C = A \oplus B \Rightarrow R_{31}=R_{11}\oplus R_{21}$

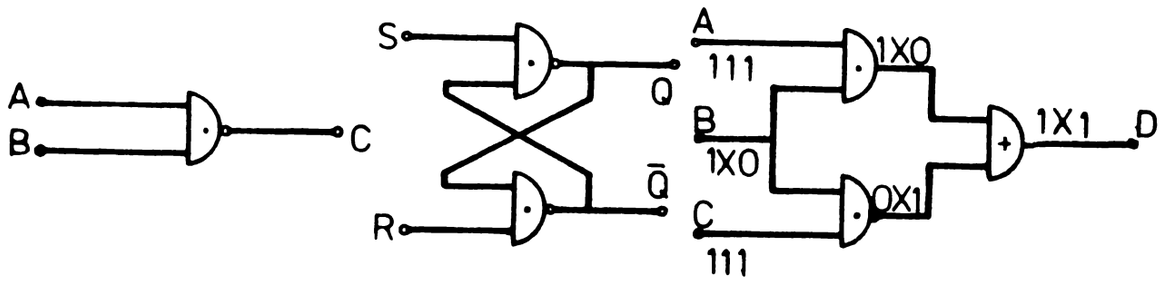
$R_{32}=R_{12}\oplus R_{22}$

Fig.3.14.

se poate stabili exact, verificarea utilizării corecte a ieşirilor legate împreună. Utilizînd starea"X" se pot face două tipuri de analize în timp, care să realizeze :

- determinarea condiţiilor de intrare ilegale - apariţia lor se semnalează, la ieşire, cu starea"X"; un exemplu ar fi aplicarea, pe intrările asincrone ale unui bistabil CDB472, a unor semnale cu valoare logică "0", adică R=S=0;
- determinarea tranziţiilor simultane pe intrările aceleiaşi componente - se semnalează, la ieşire, cu starea"X".

In concluzie, starea"X" evidenţiază orice stare a cărei valoare nu se poate determina exact.



A	B	C
0	1	1
X	X	X
1	0	1

Hazard 1X1
a)

R	S	Q	$\bar{Q}$
0	0	1	1
X	X	X	X
1	1	X	X

Hazard 1XX
b)

A	B	C	D
1	1	1	1
1	X	1	X
1	0	1	1

Hazard 1X1
c)

Fig.3.15.

In fig.3.15 se prezintă modul de apariţie şi de detecţie a hazardului cu ajutorul a trei exemple : a) în cazul unei componente SI-NU cu două intrări; b) în cazul unui bistabil RS realizat cu porţi SI-NU; c) în cazul unui circuit mai complex. Simulînd numai stările "1" şi "0" nu s-ar fi detectat hazardul. Se observă că răspunsul circuitelor este acelaşi, înainte şi după tranziţie, momentul tranziţiei fiind semnalat prin starea"X".

Chiar dacă prin simularea unei stări suplimentare (starea "X"), timpul de lucru creşte, avantajele obţinute impun acest mod de lucru.

Existenţa în biblioteca de circuite şi a timpului de propagare, caracteristic fiecărei componente, permite efectuarea unei analize exacte, în funcţie de timp, cu evidenţierea hazardu-

lui şi a impulsurilor parazite. Aceasta, spre deosebire de simulatoarele ce lucrează cu timpi de întîrziere nuli sau unitari. Practic, se poate aprecia că se lucrează cu întîrzieri variabile, element de semnalat, deoarece oferă o imagine veridică şi asupra timpului de propagare pe întreaga schemă, ceea ce este, în ultimă instanţă, precizarea frecvenţei de lucru la care ar putea fi utilizată schema de pe plachetă.

Tot cu ajutorul stării "X" se verifică şi se semnalează dacă intrările de date ale componentelor secvenţiale sincrone se modifică pe fronturile sau pe palierul (cînd este interzis) semnalului de tact.

Starea "Z" se introduce pentru a permite evaluarea corectă a ieşirilor legate împreună. Aceasta nu se poate transmite printr-un circuit logic şi de aceea este convertită în valorile "1", "0" sau "X", înainte de operaţia de propagare. Conversia se realizează cu ajutorul unei rutine, care calculează starea ieşirii utilizînd tabelul de operare din fig.3.16, în care A şi B reprezintă cei doi operanzi (ieşirile legate împreună), iar R, valoarea logică a ieşirii, după conversie şi are expresia :

$$R = A.B + "X" \cdot (A+B)$$

B \ A	0	1	X	Z
0	0	X	X	0
1	X	1	X	1
X	X	X	X	X
Z	0	1	X	Z

Fig.3.16.

Justificarea acestui mod de operare rezultă din următorul exemplu. Se presupune că există două memorii cu ieşirile, de tipul cu trei stări, legate împreună. La funcţionare normală, memoria selectată va determina starea ieşirii, deci rutina de conversie a stării "Z" va afecta ieşirii, starea logică "0" sau "1", corespunzătoare conţinutului memoriei selectate şi adresate, situaţie identică cu cazul real. Intr-un caz de defect, ambele memorii pot fi selectate simultan. In această situaţie, cazul real va determina la ieşire, o valoare, în tensiune, cuprinsă între

valorile corespunzătoare stărilor "0" şi "1". In acest caz, răspunsului rutinei de conversie i se va atribui starea "X". Deci se observă că pentru orice valori date la intrare, răspunsul rutinei va avea una din stările logice "0", "1" sau "X". Astfel se asigură interacţiunea corectă a tuturor configuraţiilor,reprezentînd orice valori logice.

In faza de exploatare se simulează circuitul corect şi cel obţinut prin injecţia unui defect. Un model al circuitului defect conţine un singur defect de tipul f0 sau f1, într-unul din nodurile schemei. In evaluarea răspunsurilor, pentru diferiţi vectori de testare aplicaţi la intrare, se calculează ieşirile numai la acele componente la care s-au modificat semnalele aplicate pe intrări sau s-a epuizat timpul de propagare, marcat de la ultima tranziţie. In felul acesta se obţine o viteză de lucru sporită, apreciindu-se statistic, că un număr redus de componente din configuraţia schemei îşi schimbă intrările la aplicarea unui nou vector de testare.

In timpul simulării se efectuează un control riguros al timpului. Orice eveniment poate apare numai la momente discrete de timp, fixate cu ajutorul unui contor t, a cărui rată de creştere Δt, reprezintă cuanta de variaţie a timpului. Un eveniment defineşte operaţia de evaluare a unei ieşiri sau a unui nod. Rezultatul evaluării poate fi sau nu, identic cu vechea valoare. Simularea începe cu operaţia de iniţializare, prin care se urmăreşte stabilirea stării fiecărui nod. Dacă acest lucru nu este posibil,i se atribuie starea "X". In continuare se pot preciza valorile fixe ale unor noduri, cum ar fi : alimentări, noduri legate la "1" sau "0" logic, prin construcţie, defecte injectate de tipul f1 sau f0, nivele constante,aplicate pentru o anumită durată de timp.

Valoarea curentă a unui semnal ce caracterizează starea unui nod este notată cu variabila VC. Ieşirea unei componente se determină, plecînd de la valoarea curentă a semnalelor aplicate pe intrările sale şi este funcţie de starea sa curentă, pentru tipul secvenţial şi nu este influenţată de starea curentă, în cazul celor combinaţionale. Noua valoare calculată este notată cu variabila VN. Aceasta poate fi programată să devină valoare curentă, după ce se epuizează timpul de propagare prin componenta respectivă. Toate evenimentele ce se află în această situaţie se introduc într-o linie de aşteptare, iar valoarea evenimentului

din capul liniei este deţinută şi în variabila VP. Linia de aşteptare va conţine într-un tablou, LA, lista evenimentelor, împreună cu momentele cînd acestea sînt programate să se producă, momente stabilite în corelaţie directă cu timpul de propagare prin componente. Un eveniment care s-a produs va avea, în rubrica timpului, valoarea zero. Algoritmul de evaluare al ieşirilor se prezintă în fig.3.17.

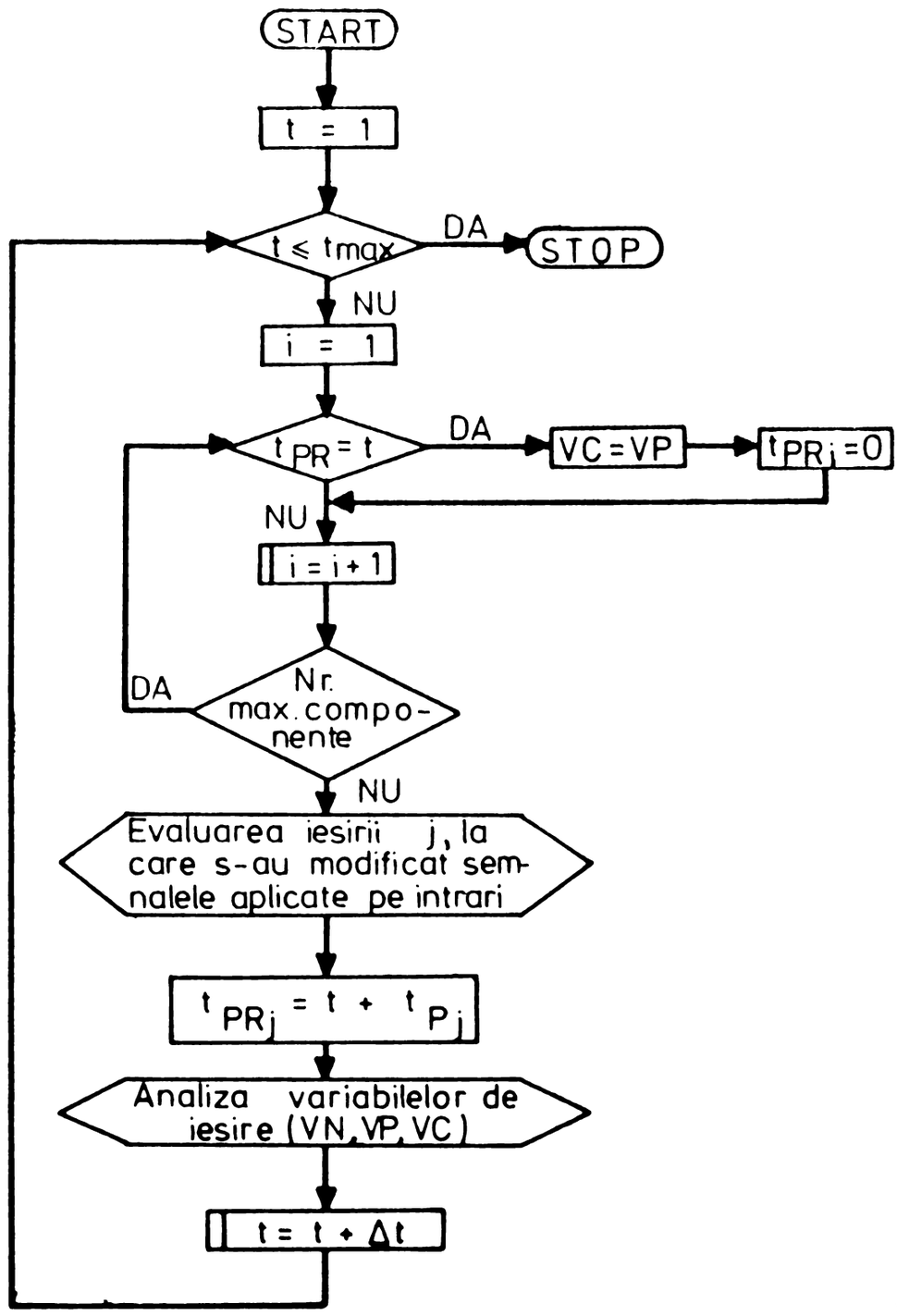


Fig.3.17.

Evaluarea unei ieşiri se face în felul următor. Dacă pentru un eveniment i, al cărui timp programat de apariţie este $t_{PR} = 0$, avem relaţia $VN \neq VC$, atunci acest eveniment va fi introdus în linia de aşteptare, iar valoarea lui, în VP. Se determină $t_{PR} = t+t_p$, în care : t - timpul curent, iar t_p - timpul de propagare prin componenta respectivă. Procedura de lucru se prezintă pe exemplul din fig.3.18.

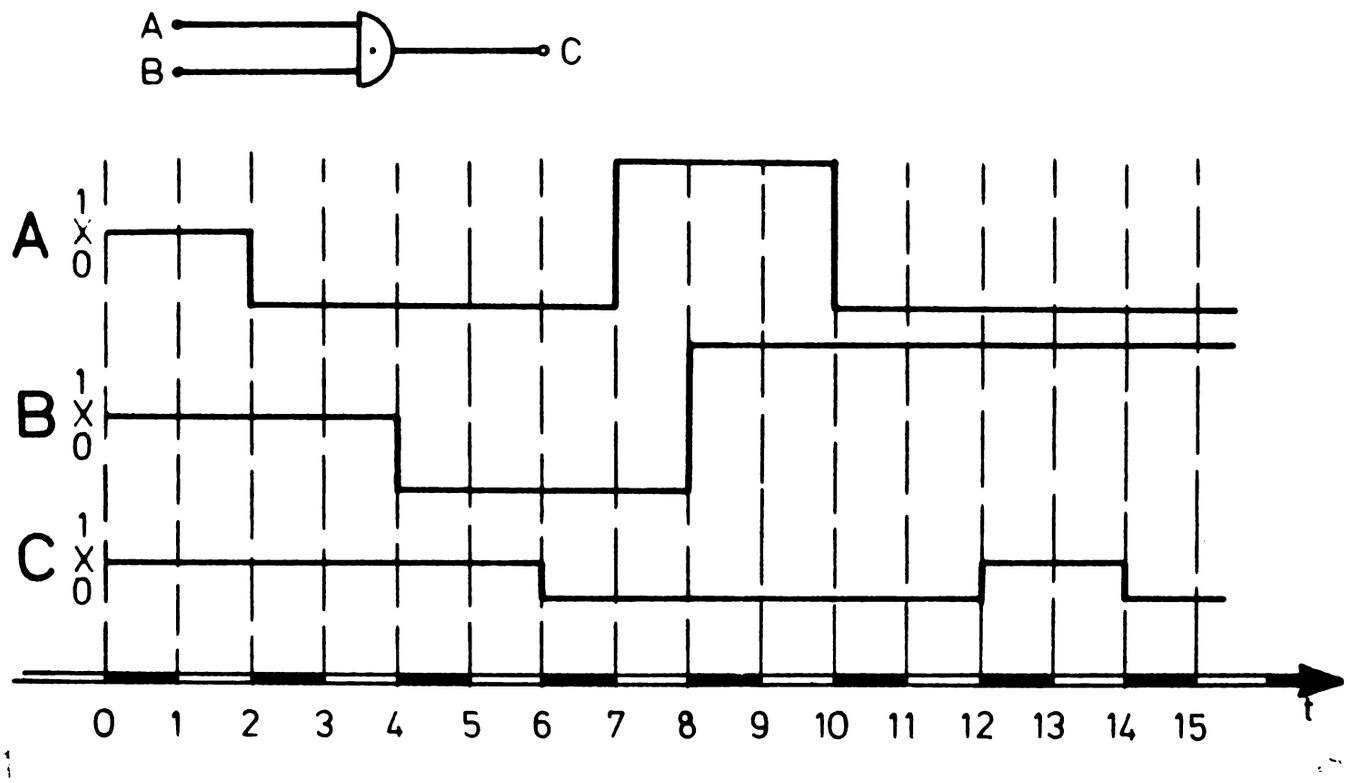


Fig.3.18.

Se presupune că poarta C are o întîrziere în propagare egală cu patru unităţi. La momentul t = 0, starea intrărilor este "X" şi deci şi a ieşirii. La momentul t = 2, pe intrarea A se aplică un semnal cu valoarea logică "0", ceea ce va solicita evaluarea ieşirii. După evaluare, variabilele de lucru aferente ieşirii C vor avea valorile : VC = X, VN = 0, VP = 0, $t_{PR} = 2+4 = 6$. La momentul t = 4, intrarea B se schimbă, trecînd în "0", ceea ce va solicita o nouă evaluare a ieşirii. Deoarece $t \neq t_{PR}$, ieşirea nu este într-o stare stabilă. Însă, efectul asupra ieşirii este acelaşi cu cel produs de modificarea intrării A, la momentul t = 2. Deci, în acest caz, nu se solicită o nouă analiză a ieşirii. Se specifică faptul că nu poate apare un impuls parazit la ieşire, nefiind o situaţie de hazard. După momentul t = 6,

ieşirea se poate considera stabilă, valoarea curentă devine cea programată, iar t_{PR} se forţează la zero, deci VC = VP, t_{PR} = 0. La momentul t = 7, intrarea A devine din nou "1" dar, deoarece ieşirea nu se modifică, nu se face nici o analiză suplimentară. La momentul t = 8, intrarea B devine şi ea "1". Aceasta va determina schimbarea valorii ieşirii din "0" în "1", schimbare care se încheie la t = 12. Valorile variabilelor de lucru vor fi : VC = 0, VN = 1, VP = 1, t_{PR} = 8+4 = 12. Dacă la momentul t = 10, intrarea A devine "0", ieşirea este influenţată negativ, deoarece este în tranziţie pînă la momentul t = 12. In această situaţie, la ieşire ar putea apare un impuls parazit, ce are influenţe nedorite asupra funcţionalităţii circuitului. După acest moment, VN = 0. Evidenţierea hazardului se face cu condiţionările : $VN \neq VP$ şi $t_{PR} \neq t$.

Se observă deci, că acest mod de evaluare permite determinarea stării oricărui nod, în orice moment discret al timpului, specificîndu-se, totodată, şi situaţiile în care pot apare impulsuri parazite, comutaţii false etc.

Dacă se simulează în acelaşi timp cu modelul corect şi un model defect, în fiecare moment solicitat, se evaluează ambele, iar ieşirile se compară între ele, semnalîndu-se cazurile de neconcordanţă. Se precizează faptul că, intrările ambelor modele sînt atacate cu aceeaşi secvenţă de stimuli. In urma analizei răspunsurilor se trage concluzia că defectul injectat se pune sau nu în evidenţă cu ajutorul secvenţei de stimuli, deci dacă aceasta poate fi sau nu utilizată ca vector de testare.

O atenţie deosebită se acordă evaluării ieşirilor circuitelor secvenţiale sincrone. In acest caz, problema care apare este cea legată de momentul basculării, corelat cu modificarea intrărilor de date. La circuitele care comută pe frontul anterior, evaluarea ieşirii se face după dispariţia frontului, deci cînd semnalul de tact are valoarea logică "1". Se semnalează ca şi eroare, modificarea semnalelor aplicate pe intrările de date, pe durata frontului anterior. La circuitele care comută pe frontul posterior, evaluarea ieşirii se face după dispariţia acestuia, deci cînd semnalul de tact are valoarea logică "0". Se semnalează ca şi eroare, modificarea semnalelor pe intrările de date, pe durata fronturilor sau a palierului semnalului de tact.

3.9. Generatorul programelor de testare

Odată cu creşterea complexităţii plachetelor echipate, construirea manuală a vectorilor de testare devine un proces tot mai ineficient şi mai greu de realizat.

Generatorul programelor de testare este destinat elaborării unui program complet, pentru întreaga schemă realizată pe plachetă, permiţînd, totodată, şi verificarea eficacităţii acestuia.

Modul de lucru se prezintă în fig.3.19. Descrierea schemei realizată pe plachetă o face utilizatorul, în ordinea dorită. Pentru construirea vectorilor de testare, în cazul fiecărei ieşiri a schemei, se apelează la unul din algoritmii aflaţi în BPVT. Pentru calculul valorii ieşirilor, în situaţia în care pe intrări se aplică semnalele stabilite prin vectorii de testare, se apelează procesorul de simulare.

In vederea evaluării eficacităţii programului construit se activează procesorul de defecte, care injectează cîte un defect în modelul descris în memoria echipamentului de calcul, se simulează modelul corect şi cel care conţine respectivul defect, aplicînd, la intrări, semnalele stabilite prin vectorul de testare, destinat acestui defect. In urma analizării rezultatelor obţinute se poate stabili eficienţa fiecărui vector de testare, în cazul prezenţei oricărui defect de tipul f0 şi f1.

Generatorul permite şi introducerea manuală a unor vectori de testare sau modificarea celor existenţi. In felul acesta se poate creşte eficacitatea programului, în detectarea şi mai ales, în localizarea defectelor. In faza de testare propriu-zisă, posibilitatea modificării vectorilor de testare, în funcţie de rezultatele obţinute în paşii precedenţi, deci realizarea unei testări adaptive, reprezintă un avantaj important pentru localizarea defectelor de pe placheta echipată.

In situaţia în care, într-un anumit nod al schemei se injectează un defect de tipul f1 sau f0, toate ieşirile componentelor se evaluează, ţinînd cont de valoarea logică a acestui nod şi indiferent de valoarea care ar trebui să o ia respectivul nod, datorită semnalelor aplicate pe intrări.

Se poate solicita afişarea vectorilor de testare, a defectelor care se detectează cu fiecare vector de testare, starea logică a diferitelor noduri, configuraţia unor tabele utilizate în

faza de simulare.

Generatorul mai contine unele comenzi utilizate în faza de exploatare, care permit realizarea a diferite facilităţi legate de activarea unor rutine din BPVT sau BIC, a unor moduri de afişare, accesul la diferitele module ale programului etc.

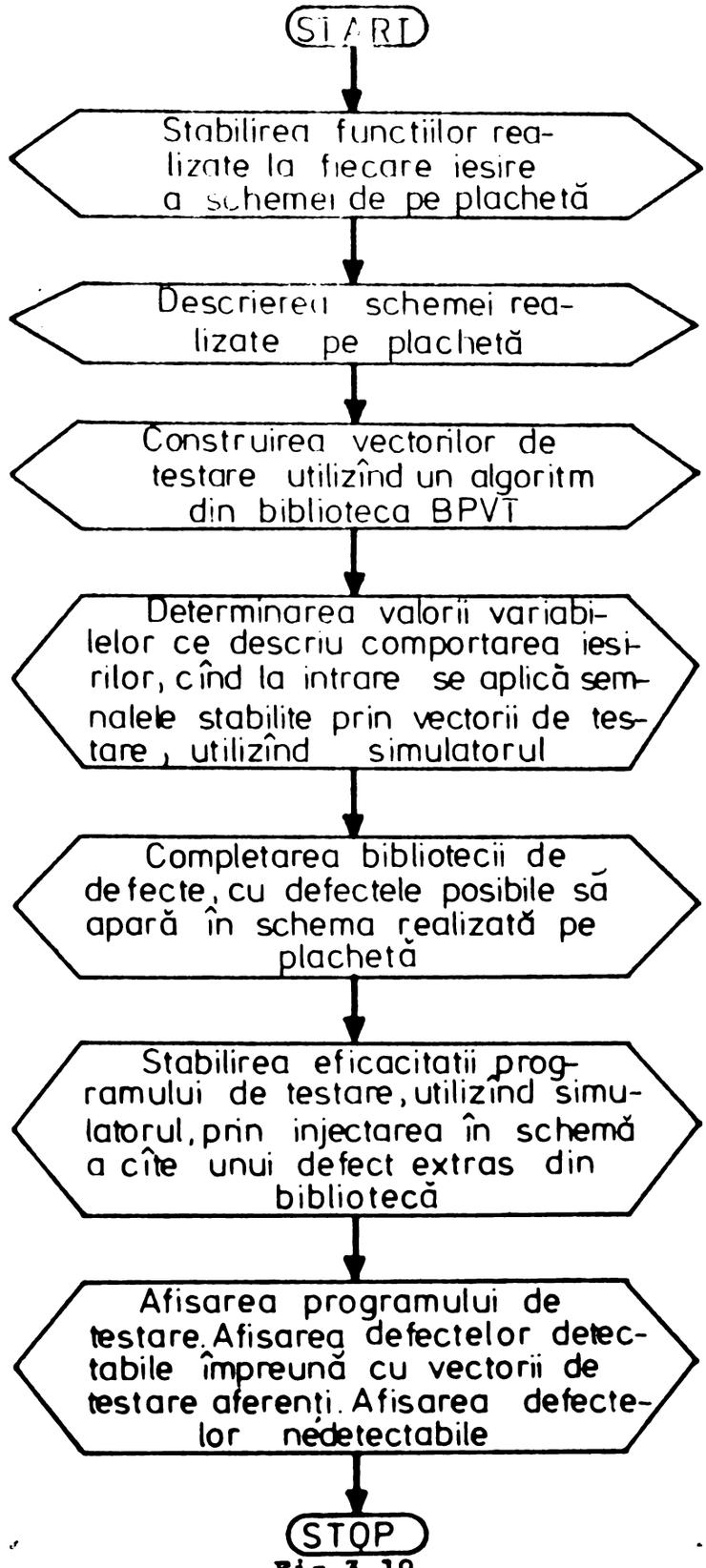


Fig.3.19.

3.1o. Concluzii

Problemele tratate în acest capitol permit enunţarea următoarelor concluzii :

a) Creşterea complexităţii circuitelor logice impune, tot mai mult, utilizarea de tehnici automate pentru stabilirea vectorilor de testare.

b) Elaborarea unui sistem de generare automată a programelor implică eforturi deosebite şi solicită sisteme de testare puternice. Gradul de detectabilitate a defectelor este cuprins între 6o-loo%, în funcţie de schema care se abordează.

c) Simulatorul propus poate fi utilizat atît în faza de testare propriu-zisă, pentru localizarea rapidă a defectelor, detectarea hazardului etc., cît şi pentru verificarea eficacităţii programelor de testare utilizate.

d) Se permite simularea simultană a două modele ale circuitului cu care este echipată placheta şi anume, cel corect şi cel în care s-a injectat un defect de tipul f0 sau f1.

e) Pentru stabilirea setului de defecte s-au definit clase de echivalenţe, care reduc considerabil numărul total de defecte.

f) Generarea programelor se face pe baza algoritmului propus în cap.2.

g) In cazul programelor de testare întocmite manual, simulatorul poate fi utilizat pentru completarea valorii ieşirilor corecte.

In continuare se pot dezvolta următoarele probleme :

a) Simularea, în paralel cu modelul corect, a unui număr mai mare de modele de defect. In acest caz creşte foarte mult eficacitatea în exploatare. Dar problema este indicat să se abordeze pe un calculator mai puternic decît M18, pe care s-a lucrat pînă în prezent. Numărul de modele defecte, simulate pe un pas, poate fi specificat de utilizator, dar are influenţă mare asupra vitezei de calcul şi a capacităţii de memorare solicitată. Se apreciază că în acest mod de lucru, totdeauna, unul dintre defecte va fi simulat cel mai bine.

b) Stabilirea de noi clase de echivalenţe a defectelor, mai ales în cazul circuitelor LSI/VLSI.

c) Utilizarea de tehnici mai eficace de evaluare a ieşirilor.

Cap.4. SISTEMUL AUTOMAT DE TESTARE

4.1. Consideraţii generale

Metodologia prezentată în capitolele doi şi trei permite construirea vectorilor de testare pentru plachetele echipate cu circuite logice. Aplicarea semnalelor electrice, descrise prin vectorii de testare, asupra plachetei ce se verifică, în configuraţia solicitată de aceasta şi înregistrarea răspunsurilor ce urmează a fi analizate, se realizează cu ajutorul unui sistem automat de testare.

Un astfel de sistem prezintă următoarele capabilităţi [N4] [*7],[*12],[*17],[*18],[*20] :

- testarea în condiţii funcţionale, cu posibilităţi de detectare şi localizare a defectelor;
- utilizarea limbajelor orientate spre testare;
- utilizarea tehnicii de simulare;
- posibilitatea construirii automate a vectorilor de testare;
- urmărirea şi vizualizarea simultană a informaţiilor furnizate pe un mare număr de căi;
- posibilităţi de adaptabilitate rapidă pentru diferitele tipuri de plachete, ce urmează a fi testate;
- realizare modulară.

In configuraţia unui sistem automat de testare se pot deosebi următoarele unităţi funcţionale, prezentate în fig.4.1 [N4] :

- modulul de testare, MT;
- modulul de atac/sesizare, MAS;
- modulul adaptor, MA;
- modulul de comandă, MC;
- echipamentul de calcul, EC.

Modulele MT, MAS, MA şi MC formează echipamentul de testare, ET, iar împreună cu echipamentul de calcul, EC, constituie sistemul automat de testare, SAT.

Modul de realizare al acestor unităţi funcţionale este determinat de tipul şi destinaţia sistemului automat de testare.

Modulul de testare are sarcina de a pregăti secvenţele de

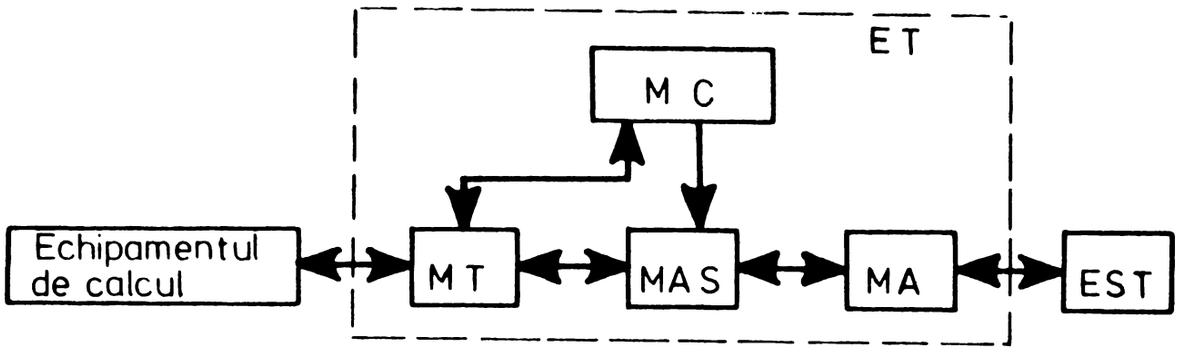


Fig.4.1.

semnale,în formatul şi configuraţia cerută de elementul care se testează (EST),precum şi de a prelua şi, eventual prelucra, rezultatele testării.

Modulul de atac/sesizare asigură aplicarea secvenţelor de semnale asupra elementului care se testează, la nivelul energetic cerut de acesta, sesizarea şi transferul răspunsurilor în modulul de testare.

Conectarea mecanică şi electrică a elementului care se testează, la sistemul de testare, se realizează prin intermediul modulului adaptor.

In fine, modulul de comandă asigură generarea semnalelor de comandă şi control necesare pregătirii şi desfăşurării operaţiei de testare, după specificaţiile utilizatorului. Este necesar acest modul, deoarece echipamentul de calcul nu poate prelua toate funcţiile de comandă şi control,la parametrii ceruţi de operaţia de testare. In primul rînd, este vorba de realizarea funcţiilor de comandă la frecvenţă mare (5oo KHz - 2o MHz).

Echipamentul de calcul este utilizat, în principal, pentru construirea vectorilor de testare şi interpretarea rezultatelor obţinute în urma testării.

Realizarea unui sistem de testare performant solicită o îmbinare perfectă între capabilităţile unui echipament de calcul puternic şi posibilităţile oferite de echipamentul de testare, legate de stimularea elementului sub test şi înregistrarea răspunsurilor acestuia.

Pentru utilizarea tehnicilor de simulare şi construire automată a vectorilor de testare, prezentate în capitolele precedente, se cere un echipament de calcul cu următoarele caracteristici : memoria operativă 64-256 K, lungimea cuvîntului 8-16 biţi, facilităţi multiple de intrare/ieşire, set puternic de instruc-

ţiuni, 1-2 unităţi de discuri, consolă (DAF). Aceste trăsături le posedă, în parte, microcalculatorul M118 şi, mai ales, minicalculatorul CORAL 4oll [*47],[*48].

Asupra elementelor caracteristice ale echipamentului de testare se va insista amănunţit în paragrafele următoare.

Intr-o operaţie de testare se pot analiza parametrii staţiei, dinamici şi/sau funcţionali.

Tipul sistemului de testare este definit, în principal, de destinaţia acestuia. In acest sens se elaborează sisteme distincte pentru testarea de : componente active şi pasive, circuite integrate SSI, MSI şi LSI, plăchete neechipate, sertare, cable, plachete echipate şi produse finale.

Sistemele destinate testării componentelor pasive [* 4], [*37],[*41],[*42] verifică rezistenţe, condensatoare, inductanţe. In general, se realizează configuraţii mai simple, comparativ cu alte tipuri de sisteme automate de testare. Rolul modulului de comandă este preluat, de cele mai multe ori, de către echipamentul de calcul, realizat, în majoritatea cazurilor, pe baza unui microprocesor. Aceasta este posibilă datorită vitezei mici de prelucrare solicitată. Nu necesită un limbaj orientat spre testare, diferitele comenzi legate de modurile de lucru şi tehnicile utilizate fiind activate de la un panou sau prin explorarea conţinutului unei memorii fixe. Este posibilă selectarea frecvenţei şi amplitudinii semnalelor care se aplică asupra componentei ce se testează, în funcţie de tipul şi valoarea acesteia, măsurătoarea efectuîndu-se, fie în configuraţii seriale, fie paralele.

Sistemele de testare destinate componentelor active sînt foarte diverse, după tipul componentelor şi gama de parametri care se verifică. Cele mai complexe sînt sistemele de testare hibride, care pot analiza atît componente logice cît şi analogice, urmărind evaluarea parametrilor statici, dinamici şi funcţionali [*39],[*41],[*42],[*43].

In fig.4.2 se prezintă un sistem automat destinat testării componentelor active.

Schimbul de informaţii cu echipamentul de calcul se realizează prin interfaţa IIE. Decodificatorul de adrese, DADR, permite activarea oricărui bloc al modulelor MT şi MAS. Programarea stimulilor de tip intensitate şi tensiune se asigură cu ajutorul blocurilor PSC, respectiv PSV, care au în construcţia lor convertoare numeric-analogice. Generarea acestor stimuli se face în

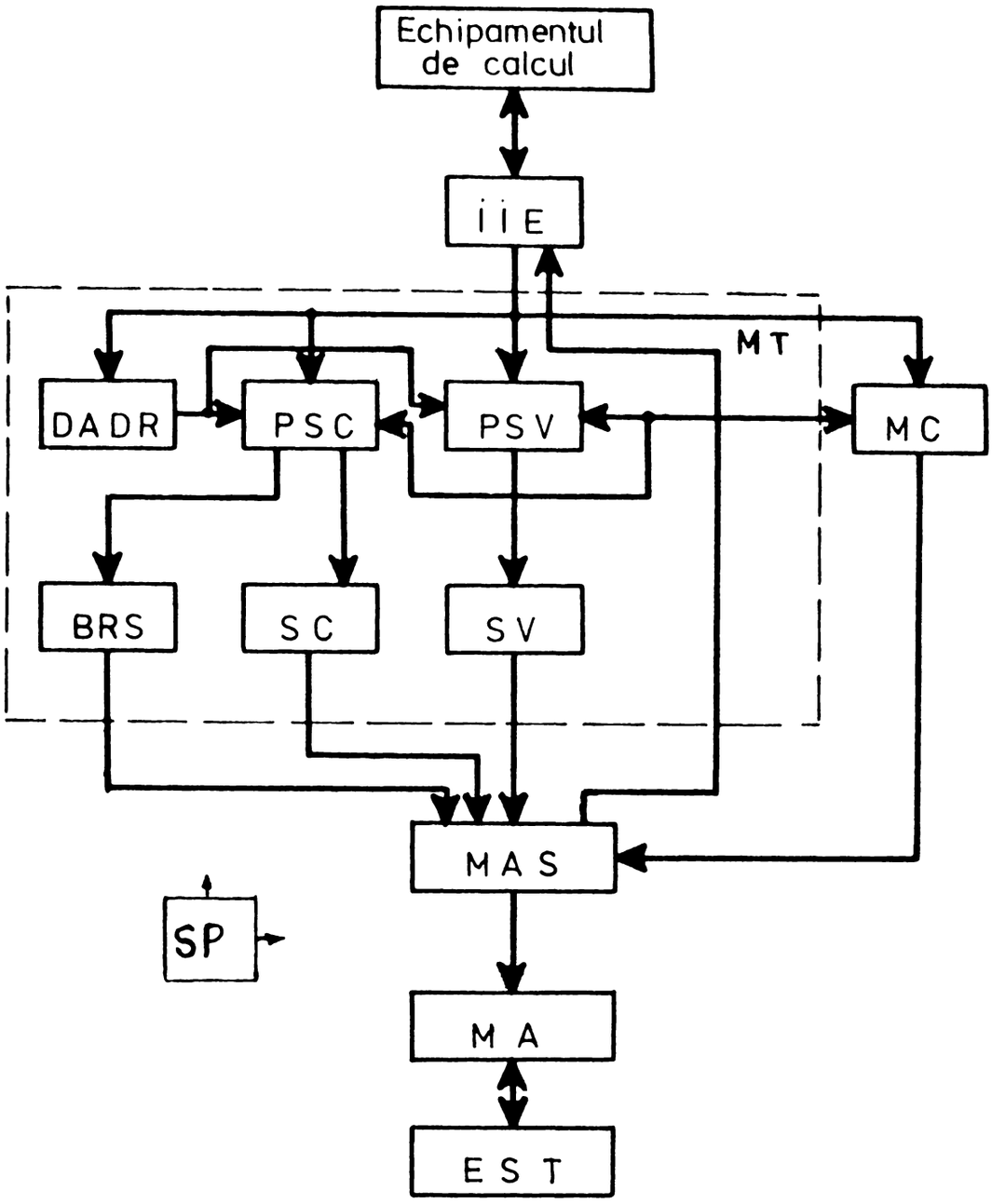


Fig.4.2.

blocurile SC, respectiv SV. Testarea în sarcină se realizează cu ajutorul blocului rezistenţelor de sarcini, BRS, care se programează prin intermediul blocului PSC. Ansamblul surselor programabile, SP, alimentează atît echipamentul de testare cît şi componenta care se testează.

Cu ajutorul modulelor MAS şi MA se pot conecta mai multe staţii de testare.

Sistemele destinate testării circuitelor integrate [*11],[*22],[*35],[*37],[*41],[*2],[*3],[*44] urmăresc realizarea, în primul rînd, a unei verificări parametrice, în condiţii reale de exploatare. Testarea parametrilor funcţionali se efectuează pe baza tabelelor de adevăr, memorate.

In cazul circuitelor mai complexe (memorii, microprocesoare etc.) se utilizează diferiţi algoritmi specifici, pentru construirea vectorilor de testare (cum este cel prezentat în cap.2).

Sistemele destinate testării plachetelor neechipate, a sertarelor şi a cablelor [*37],[*41],[*42] urmăresc, în principal, verificarea continuităţii electrice între două puncte. Astfel se detectează legăturile lipsă şi scurtcircuitele existente, măsurînd rezistenţa traseului între două puncte. Pentru valori maxime de ordinul ohmilor se poate aprecia existenţa unui scurtcircuit, iar pentru valori peste lo K, lipsa legăturii. Probleme deosebite, în cazul acestor sisteme, apar la realizarea modulelor MAS şi MA, datorită multitudinii punctelor care trebuie verificate. Se pot utiliza diferiţi algoritmi pentru minimizarea timpului de testare. Rolul modulelor MC şi MT poate fi preluat, în mare parte, de către echipamentul de calcul, care printr-o interfaţă de intrare/ieşire poziţionează şi realizează măsurătorile, direct ,cu ajutorul modulului MAS şi a capetelor de acces, din modulul MA.

Sistemele destinate testării plachetelor echipate cu circuite logice [*4],[*5],[*6],[*7],[*12],[*13],[*17],[*18],[*2o], [*23],[*27],[*28],[*35],[*37],[*38],[*4o] sînt unele dintre cele mai complexe, atît din punct de vedere al configuraţiei, cît şi al sistemului de programe necesar. Totodată, cu aceste sisteme se poate aborda testarea şi a altor tipuri de elemente (cum ar fi, de exemplu, componentele logice integrate), fără ca modificarea destinaţiei să ridice, în general, probleme deosebite.

Aceste consideraţii, ţinînd cont şi de importanta utilitate practică, justifică prezentarea ,în paragrafele următoare, a unui sistem automat destinat testării plachetelor echipate cu circuite logice SSI, MSI şi LSI, conceput de autor.

Sistemele de testare pentru produse finale simulează modurile de funcţionare ale acestora. Procedurile de lucru sînt particulare fiecărui tip de produs şi asigură o testare a parametrilor funcţionali. Deci acestea sînt sisteme orientate, fără să aibă o aplicabilitate generală.

4.2. Configuraţia sistemului automat de testare a plachetelor echipate cu circuite logice SSI, MSI şi LSI [*2],[*43],[*5],[*7],[*12],[*17],[*2o],[*23],[*27], [*29],[*31],[*32],[*34],[*38]

Sistemul ce se prezintă este destinat testării funcţional-

dinamice, în regim de detectare şi localizare a defectelor, pe plachetele echipate cu circuite logice SSI, MSI şi LSI, aparţinînd familiilor logice TTL şi CMOS. Fiind realizat într-o configuraţie modulară, orientată pe magistrala universală IEEE 488 - 1975, sistemului i se pot conferi capabilităţile dorite, în funcţie de natura practică a aplicaţiei în care se utilizează. Modul de realizare şi performanţele ce le asigură sînt proprii generaţiei 3,5, aferentă acestui domeniu.

Pentru obţinerea unui mod de interfaţare şi de schimb a informaţiilor, universal, indiferent de tipul modulului, conceptul de magistrală universală se utilizează pentru toate modulele echipamentului de testare, indiferent de tip, aceasta spre deosebire de sistemele de testare existente, care schimbă prin magistrală numai informaţiile aferente modulelor analogice. Totodată, se precizează faptul că sistemul de programe de bază este puţin afectat, în comparaţie cu situaţia cînd există mai multe moduri de schimb a informaţiilor. Viteza globală de transfer rămîne, în continuare, ridicată. Toate aceste avantaje se obţin în detrimentul unei creşteri a numărului de circuite utilizate,cu aprox. (1o-3o) buc.,pentru o interfaţă, aceasta în funcţie de complexitatea interfeţei.

Configuraţia sistemului se prezintă în fig.4.3 şi conţine următoarele elemente :

a) - modulul de testare, MT, are următoarele blocuri :

a1) - pentru aplicarea informaţiei pe intrările plachetei ce se verifică :

- separatorul de intrare/ieşire, SIE - specifică pinii de intrare şi cei de ieşire de pe plachetă;
- selectorul familiei logice, SFL - stabileşte tipul familiei logice (TTL sau CMOS) în care se încadrează funcţionarea circuitului a cărui intrare, la nivel de pin al plachetei, se specifică în instrucţiune;
- selectorul atributelor secvenţei, SAS - selectează tipul informaţiei ce se aplică pe fiecare pin de intrare. Pentru activarea circuitelor, pe pinii de intrare se pot aplica semnale de tip nivel logic sau tacte. Specificarea valorii logice a acestor semnale se face prin vectorii de testare şi, deci, modificarea acestor valori se obţine prin schimbarea

vectorilor de testare. Aceasta se poate realiza la viteza de prelucrare şi transfer a complexului echipament de calcul - magistrală sau la o viteză superioară, obţinută prin utilizarea de blocuri suplimentare pentru formarea şi/sau memorarea intermediară a vectorilor de testare;

- distribuitorul de nivele logice, DNL - permite stabilirea stimulilor de tip nivel logic din configuraţia unui vector de testare, în formatul cerut de placheta care se testează, adresarea pinilor asupra cărora se aplică aceşti stimuli şi atacul simultan al tuturor pinilor de intrare pe plachetă;
- blocul de memorie pentru păstrarea temporară a vectorilor de testare, MVT - este destinat memorării unui număr de vectori de testare, care ulterior, la o comandă, vor fi aplicaţi asupra plachetei ce se verifică, realizîndu-se astfel o succesiune de atac la viteză foarte mare, limitată superior numai de timpul de acces al memoriei;
- distribuitorul stimulilor aleatori, DSA - aceşti stimuli sînt generaţi de analizorul de semnături, AS, prin diferite tehnici heuristice. Rezultatul, culeş din nodurile plachetei cu ajutorul unor sonde, se comprimă, după o tehnică oarecare, sub forma unei "semnături", care se compară cu cea corectă;
- distribuitorul de tacte, DT - permite aplicarea pe plachetă, la pinii specificaţi, a unui număr dorit de tacte, a căror polaritate şi frecvenţă sînt programabile;

a2) - pentru citirea stării pinilor de intrare sau de ieşire ai plachetei ce se verifică :

- blocul de înregistrare al răspunsului obţinut în urma aplicării semnalelor descrise de un singur vector de testare, BRVT;
- blocul de memorie pentru înregistrarea răspunsurilor obţinute în urma aplicării semnalelor descrise de un număr dat de vectori de testare, memoraţi în blocul MVT, MRVT ;

b) - modulul de comandă al testării, MC, cuprinde :

- numărătorul de tacte, NT - permite aplicarea pe intrările de tact, a unui număr de semnale de tact precizate

prin vectorul de testare;

- numărătorul de întîrzieri, NI- precizează întîrzierea necesară din momentul aplicării semnalelor, descrise printr-un vector de testare, asupra plachetei, pînă se înregistrează răspunsul. Această întîrziere se specifică în număr de unităţi de timp, mărimea unei unităţi fiind programabilă;
- generatorul de tacte programabile, GTP - realizează programarea frecvenţei, duratei şi polarităţii semnalelor de tact;
- blocul de oprire condiţionată, BOC - determină oprirea aplicării semnalelor descrise de un nou vector de testare şi/sau înregistrarea condiţionată a unui răspuns, în urma apariţiei unui eveniment generat de : un tip de răspuns aşteptat, număr de tranziţii îndeplinit, epuizarea timpului de aşteptare programat etc. Acest bloc lucrează corelat cu blocurile MVT şi MRVT;
- panoul de testare, PAT - permite stabilirea modului de lucru şi de afişare a rezultatelor testării, precum şi emiterea unor comenzi, utilizate în operaţia de testare;
- sincronizatorul de test, ST - comandă aplicarea sincronă a semnalelor descrise într-un vector de testare, pe toţi pinii de intrare ai plachetei, precum şi programarea intervalului dintre momentul atacului şi cel al înregistrării răspunsului. In felul acesta se realizează şi o verificare a timpului de propagare a semnalelor prin circuitele plachetei. Sincronizarea se poate face, fie cu tactul intern al sistemului, fie cu un tact sau cu o comandă externă, eventual chiar culese de pe placheta care se testează;

c) - modulul de atac/sesizare, MAS;

d) - modulul de adaptare, MA.

Cuplarea la echipamentul de calcul se realizează prin :

- interfaţa de recepţie şi emisie a semnalelor, IRES;
- blocul de comandă şi control a magistralei universale,BCCM.

Schimbul de informaţii se face pe următoarele magistrale:

- magistrala bidirecţională de informaţii, MIB;
- magistrala de comenzi ale interfeţei, MCI.

Aceste două magistrale sînt caracteristice interfeţei universale IEEE 488.

- magistrala de intrare pentru date, MID;

- magistrala de ieşire a datelor , MED;
- magistrala de adrese, MA;
- magistrala de comenzi, MC.

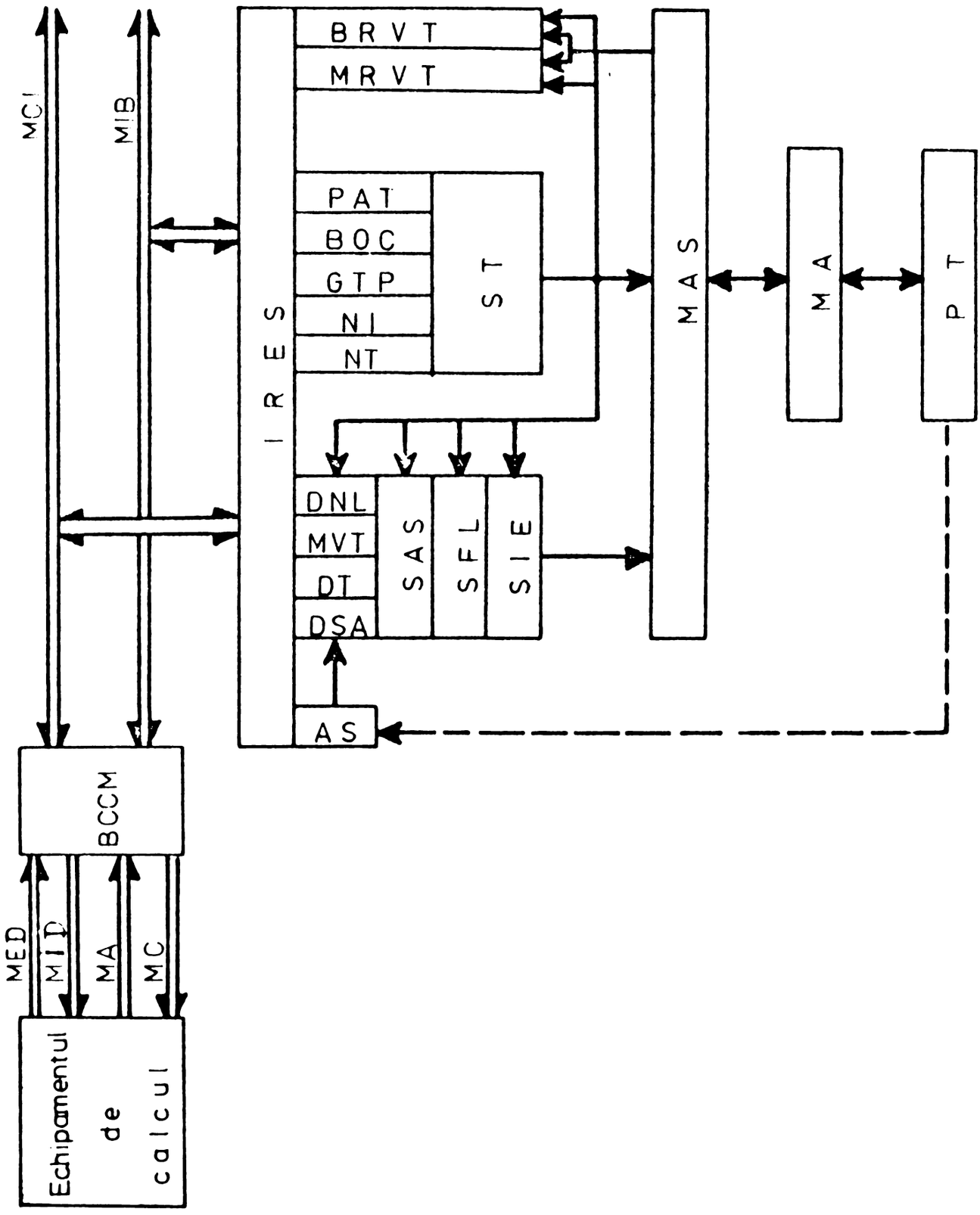


Fig.4.3.

4.3. Magistrala universală a sistemului de testare [*49], [*5o], [*51], [*52], [*53], [*4], [*5], [*1o], [*2o], [*26], [73]

Utilizarea unei tehnici de cuplare standardizate între diferitele module ale unui sistem oferă avantaje multiple atît în faza de proiectare, cît şi de punere la punct sau exploatare. O astfel de tehnică presupune existenţa unei magistrale universale standardizată, care cere la rîndul ei, interfeţe standardizate. Oferind o soluţie unică, privind interconectarea modulelor unui sistem, acest mod de lucru permite simplificarea sau chiar eliminarea unor blocuri funcţionale, contribuind, în principal, la reducerea substanţială a timpului de punere în funcţiune. Adăugarea unui modul nou sau eliminarea unuia existent nu ridică nici o problemă deosebită.

Preocupările legate de standardizare se referă la elaborarea funcţiilor interfeţelor, la modul de schimb a informaţiilor, condiţiile de prelucrare a întreruperilor şi cuprind specificaţii electrice, mecanice şi funcţionale.

Selectarea tipului de magistrală universală, pentru un sistem dat, se face luînd în consideraţie următorii parametri :

- viteza de transfer a informaţiilor;
- numărul de module care vor intra în configuraţia sistemului;
- lungimea magistralei;
- numărul de linii ale magistralei;
- complexitatea şi costul realizării;
- facilităţile ce le oferă în exploatare.

Prin funcţiile interfeţei, un modul conectat la magistrală trebuie să poată emite şi/sau recepţiona mesaje sau să controleze schimbul de informaţii.

Din motivele arătate mai sus, urmărind în final realizarea unei interfaţări universale, atît la nivel hard cît şi soft, cu un schimb de informaţii unic acceptat de toţi cei care concură la magistrală, se propune utilizarea magistralei universale, împreună cu funcţiile de interfaţare standardizate asociate, pentru toate modulele, indiferent de tip (numeric, analogic, hibrid) sau destinaţie.

In general, se pot utiliza două tipuri de magistrale : paralele şi seriale.

Din categoria celor paralele, cele mai performante sînt :
- magistrala S100;
- magistrala IEEE 488;
- magistrala IEEE 583;
- magistrale orientate pe diferite microprocesoare.

Magistralele seriale cele mai cunoscute sînt :
- EIA - RS232C - pentru comunicaţii asincrone;
- EIA - RS422-423 - pentru comunicaţii asincrone şi sincrone.

Magistrala paralelă, cu un număr de linii mult mai mare decît cea serială, se utilizează în aplicaţiile care solicită efectuarea schimbului de informaţii la viteze ridicate. Pentru viteze mici, dispunînd, însă, de o siguranţă şi fiabilitate sporite, se indică utilizarea magistralei seriale.

Magistrala S100 conţine 100 de linii, cu următoarele destinaţii : 16 linii de adrese, 8 linii pentru introducerea datelor, 8 linii pentru extragerea datelor, 8 linii de întrerupere, 39 linii de control, 3 linii de alimentare şi 18 linii la dispoziţia utilizatorului.

Primul element caracteristic, care iese în evidenţă, este complexitatea magistralei. Aceasta este un avantaj în faza de proiectare, dar pot apare probleme dificil de rezolvat legate de imunitatea la perturbaţii, mai ales la frecvenţe ridicate (peste 1 MHz), fiabilitate scăzută, exploatare raţională, punerea în funcţiune şi, chiar şi preţul de cost. Numărul liniilor de control este prea ridicat şi, în general, orientat pe modul de lucru al circuitului 8228/8238, produs de firma Intel.

Magistrala 6800 posedă : 8 linii bidirecţionale pentru date, 16 linii unidirecţionale pentru adrese şi 9 linii de control. Este o magistrală mult mai compactă, orientată pe modul de lucru al microprocesorului Motorola 6800. Poate fi utilizată pentru transferul de mare viteză. Fiind orientată pe un tip de microprocesor, este utilizată cel mai mult în sistemele ce au în configuraţia lor acest element.

Standardul IEEE 583, denumit şi CAMAC, stabileşte organizarea magistralelor, interfeţelor, alimentărilor şi caracteristicile mecanice. Acest tip de standard conţine o magistrală paralelă şi una serială. Magistrala paralelă, de viteză, are următoarele linii : 3 linii de control, 5 linii de comandă, 5 linii de adrese, 24 linii pentru introducerea informaţiilor şi 24 linii pentru extragerea informaţiilor. Specificaţiile mecanice şi

electrice sînt foarte riguroase, astfel încît să se asigure o viteză de transfer ridicată (pînă la 24 Mbit/S), cu un factor de siguranţă şi fiabilitate ridicate. Datorită acestor specificaţii, realizarea este dificilă şi costisitoare. Se utilizează, cel mai mult, în tehnica nucleară, unde se cere realizarea transferului a mari volume de date, la viteze şi siguranţă ridicate.

Standardul IEEE 488 este destinat interconectării de echipamente foarte diverse : aparate de măsură şi control, calculatoare, periferice. Dispunînd de o configuraţie generală, fără o orientare predefinită, standardul conţine specificaţii mecanice, electrice şi funcţionale. Magistrala este formată din 16 linii, repartizate astfel : 8 linii bidirecţionale pentru transferul datelor, 5 linii de comandă şi 3 linii pentru controlul transferului.

Transmisiile seriale utilizează una sau două linii, adresele, datele şi comenzile fiind transferate bit cu bit. Aceasta este motivul pentru care viteza de transfer este mai scăzută. Dar numărul scăzut de linii contribuie la obţinerea unei siguranţe ridicate, în funcţionare.

Gradul de generalitate, uşurinţa în proiectare, punerea la punct şi exploatarea, alături de o siguranţă ridicată în funcţionare, au impus standardul IEEE 488,în cele mai diverse domenii, fiind astăzi cel mai utilizat în practică.

Datorită performanţelor ce le realizează, am adoptat acest standard în configuraţia sistemelor de testare, motiv pentru care se va prezenta detaliat, în continuare. In paranteze sînt prezentate denumirile originale din standard .

Pe magistrala universală se transmit două tipuri de informaţii : mesajele interfeţei şi mesajele dependente de modul. Primele sînt destinate stabilirii funcţiilor interfeţei, fără să afecteze modulele. Acestea sînt standardizate, iar prezenţa lor pe magistrală este semnalată printr-un semnal de comandă. Al doilea tip de mesaje este destinat modulelor, fără să afecteze funcţiile interfeţelor. Un tip particular de mesaje îl constituie cele locale. Acestea circulă între module şi interfaţă.

O interfaţă standard poate realiza maximum lo funcţii, proiectantul urmînd să aleagă pe acelea care sînt solicitate de modul de lucru al modulului ce se interconectează. Totodată, pentru o funcţie stabilită, se pot reţine numai acele capabilităţi, care sînt strict necesare aplicaţiei date. In felul acesta se

poate realiza o interfaţă optimă, cu minimum de circuite, atît ca număr cît şi ca tip, viteză maximă de transfer, toate acestea orientate spre un anumit tip de modul ce se interconectează, respectînd, însă, standardul atît în ceea ce priveşte realizarea funcţiilor, cît şi a schimbului de informaţii.

Funcţiile posibile sînt următoarele : funcţia de control - C,(controller), emiţător - T,(talker), receptor - L,(listener), acceptor de informaţie - AH, (acceptor handshake), sursă de informaţie - SH, (source handshake), cerere de întrerupere - SR, (service request), lucrul în local - RL (remote local), interogare paralelă - PP,(parallel poll), iniţializare modul - DC, (device clear), start modul - DT, (devine trigger). Dintre acestea, funcţiile C, T, L, SH şi AH sînt de bază, celelalte fiind auxiliare. Oricărui modul conectat la magistrală i se poate atribui una din funcţiile : emiţător, receptor, de control sau o combinaţie a acestora. La un moment dat va fi activă o singură funcţie de control, un singur emiţător şi cel puţin un receptor. După ce funcţia C activă a transmis mesajele de comandă diferitelor interfeţe, magistrala este eliberată, astfel încît se poate realiza schimbul de informaţii între un eventual emiţător şi receptoarele active. Cu ajutorul funcţiei C se stabileşte emiţătorul şi receptoarele active la un moment dat.

Fiecare funcţie a interfeţei se descrie prin una sau mai multe grupe de stări înlănţuite, mutual exclusive. Dintr-o grupă de stări, una singură este activă la un moment dat.

Realizarea stărilor nu implică, neapărat, existenţa unor configuraţii secvenţiale (de exemplu, circuite bistabile), iar în cazul existenţei, rămîne la alegerea proiectantului modul de lucru sincron sau asincron. Pentru cazul nostru, am considerat oportun, oferind o siguranţă ridicată în exploatare, fără ca aceasta să ducă la creşterea numărului de circuite, utilizarea bistabilelor J-K MS pentru memorarea stărilor şi deci, modul de lucru sincron.

Modul de interconectare al celor 10 funcţii se prezintă în fig.4.4.

Partiţionarea interfeţelor în funcţii este esenţială pentru facilitarea procesului de realizare şi asigură un înalt grad de opţionalitate, flexibilitate şi acomodarea la diferite nivele de capabilităţi.

Pentru exemplificare se prezintă modul de realizare al

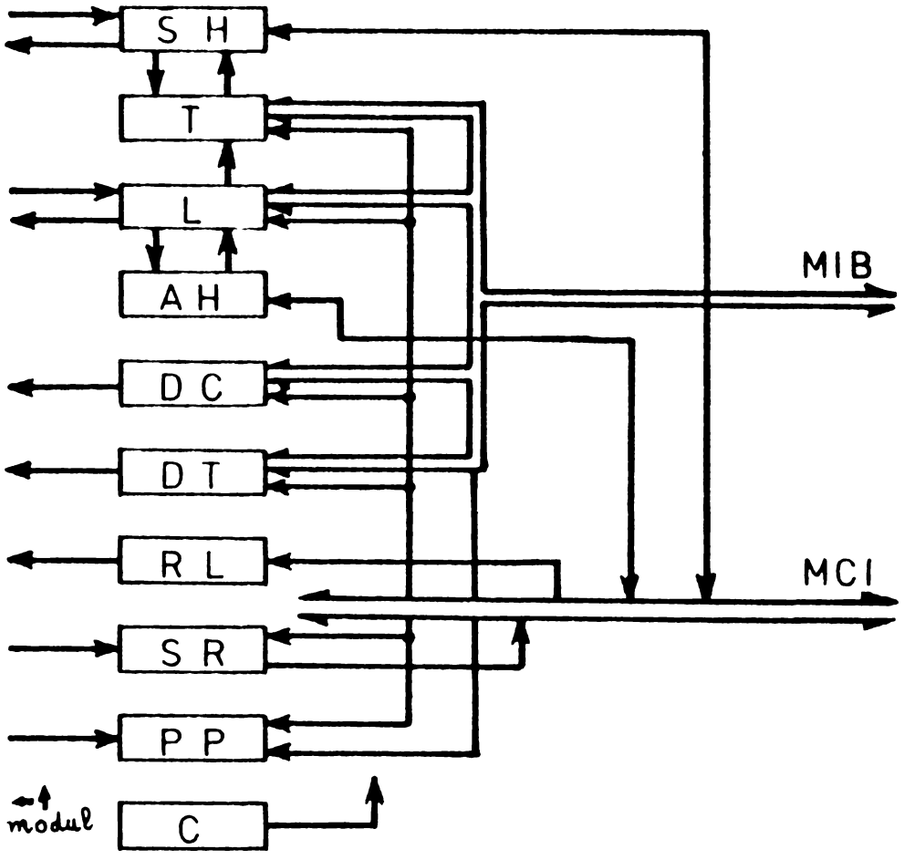


Fig.4.4.

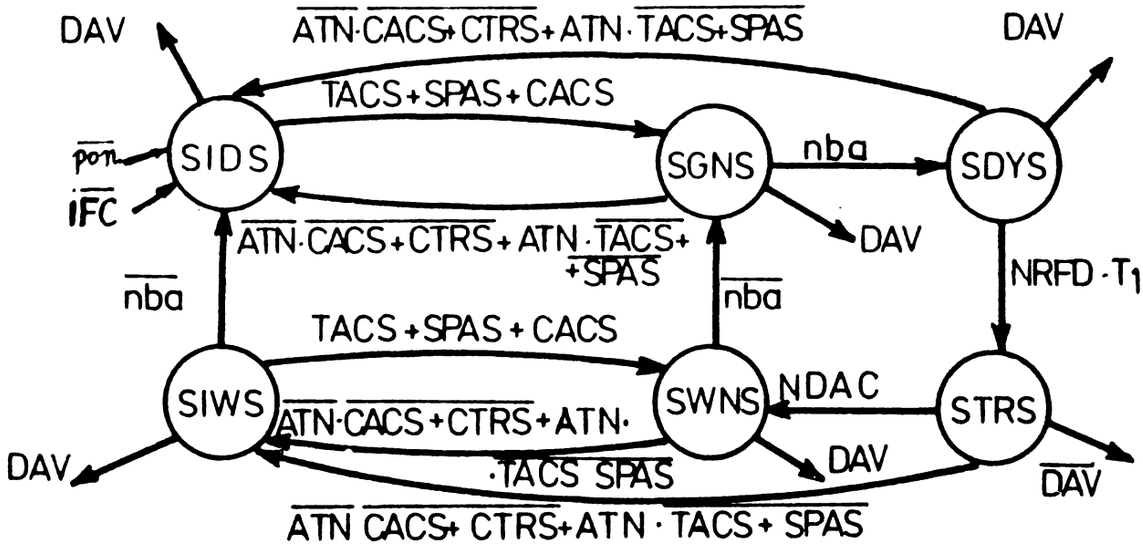


Fig.4.5.

funcţiei SH. Această funcţie dă posibilitatea unui modul să efectueze un transfer de mesaje spre unul sau mai multe module ale căror funcţii receptoare sînt active. Transferul informaţiei se face asincron, funcţia controlînd începutul şi sfîrşitul opera-

ţiei.

Diagrama de stare a funcţiei se prezintă în fig.4.5.

Se adoptă următoarea codificare a stărilor :

$SIDS = \overline{Q_2}.\overline{Q_1}.\overline{Q_0}$; $STRS = Q_2.Q_1.Q_0$;

$SGNS = \overline{Q_2}.\overline{Q_1}.\overline{Q_0}$; $SWNS = Q_2.\overline{Q_1}.Q_0$;

$SDYS = \overline{Q_2}.Q_1.Q_0$; $SIWS = Q_2.\overline{Q_1}.\overline{Q_0}$.

Ecuaţiile de funcţionare, minimizate, rezultă sub forma :

$J_2 = SDYS.NRFD.T_1$;

$K_2 = \overline{nba} \cdot Q_2 \cdot \overline{Q_1}$;

$\overline{R_2} = \overline{pon} \cdot \overline{IFC}$;

$J_1 = SGNS \cdot nba$;

$K_1 = \overline{ATN.CACS.CTRS.ATN.TACS.SPAS.Q_1.Q_0.STRS.NDAC}$;

$\overline{R_1} = \overline{pon} \cdot \overline{IFC}$;

$J_o = \overline{TACS.CACS.SPAS}.\overline{Q_1}.\overline{Q_0}$;

$K_o = Q_o.\overline{ATN.CACS.CTRS.ATN.TACS.SPAS}$;

$\overline{R_o} = \overline{pon} \cdot \overline{IFC}$.

Pe baza acestor ecuaţii se construieşte schema electrică a funcţiei, prezentată în fig.4.6.

In starea SIDS, funcţia SH nu este angajată în ciclul de dialog. Aceasta apare la punerea sub tensiune ($\overline{pon}$). In starea SGNS, funcţia aşteaptă ca noul mesaj generat de către modul să devină disponibil. In starea SDYS, mesajul se stabilizează pe liniile magistralei, iar funcţiile acceptoare urmează să semnalizeze disponibilitatea lor. In această stare se ajunge, dacă modulul lansează mesajul local nba, prin care specifică faptul că un nou cuvînt de informaţie este disponibil. Dacă funcţia AH semnalizează că poate primi un nou cuvînt, prin emiterea semnalului NRFD, funcţia SH trece în starea STRS, în care, prin semnalul DAV, anunţă funcţiile acceptoare ale modulelor receptoare active că pe magistrala de informaţii MIB este disponibil un nou cuvînt. După recepţionarea acestuia, funcţiile acceptoare determină trecerea semnalului NDAC din starea logică "0", în starea logică "1", ceea ce va avea ca efect comutarea funcţiei SH în starea de aş-

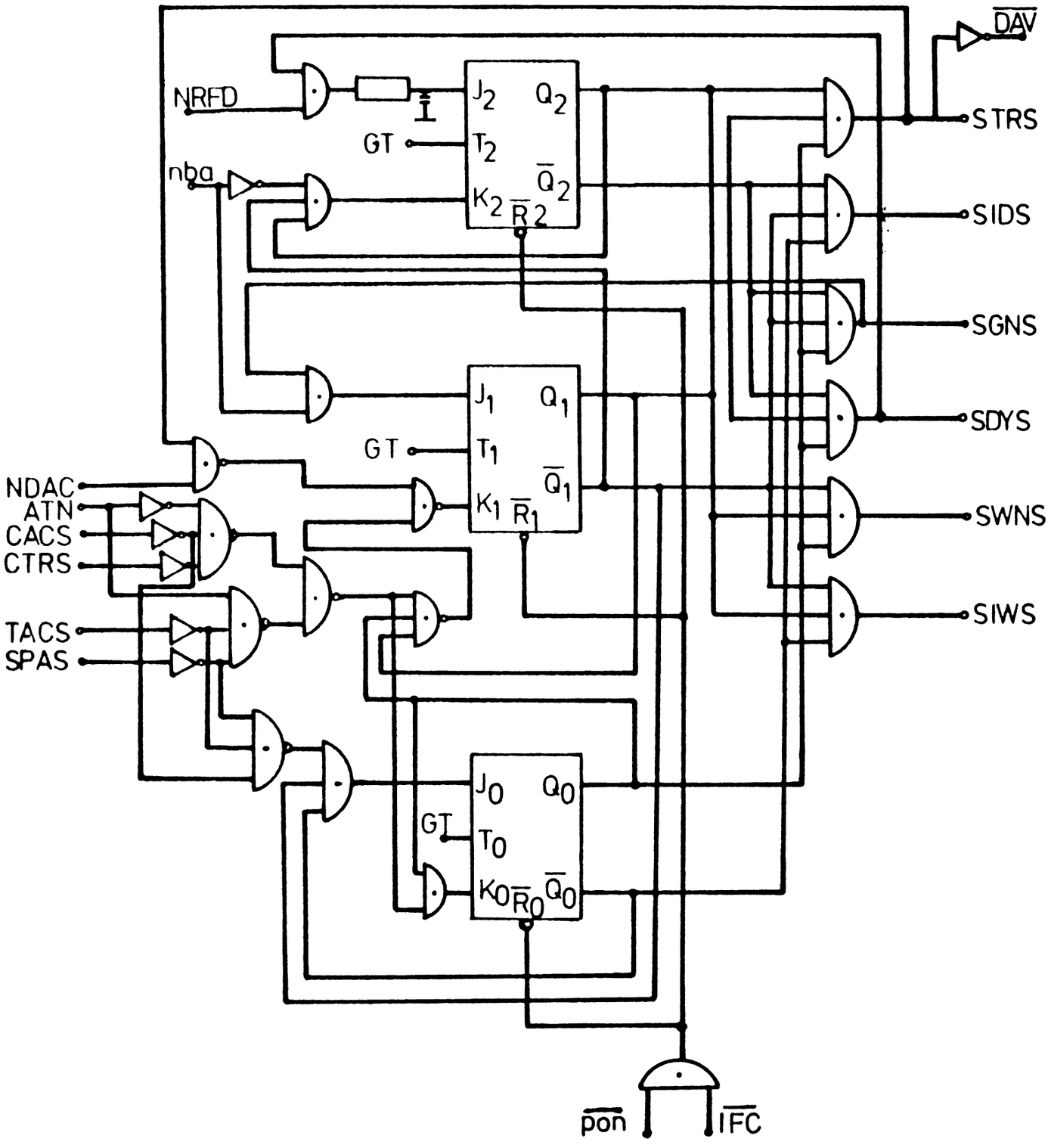


Fig.4.6.

teptare SWNS şi apoi în starea SGNS, unde aşteaptă o nouă apariţie a mesajului local nba. Starea SIWS permite întreruperea unei secvenţe de transfer, fără să se piardă informaţia din interfaţă, în timp ce modulul se pregăteşte pentru un nou ciclu de generare.

Este indicat, din motive de siguranţă în exploatare, ca mesajul nba să apară în stările SIDS sau SGNS şi poate dispare în orice altă stare.

Plecînd de la diagrama de bază din fig.4.5, se prezintă în fig.4.7, o diagramă minimizată, care reţine numai o parte din tranziţiile şi condiţionările prezente în diagrama iniţială. Aceasta minimizare nu va afecta, însă, modul de schimb al informaţiilor stabilit în standard, ci numai unele capabilităţi, care nu sînt solicitate în majoritatea aplicaţiilor practice.

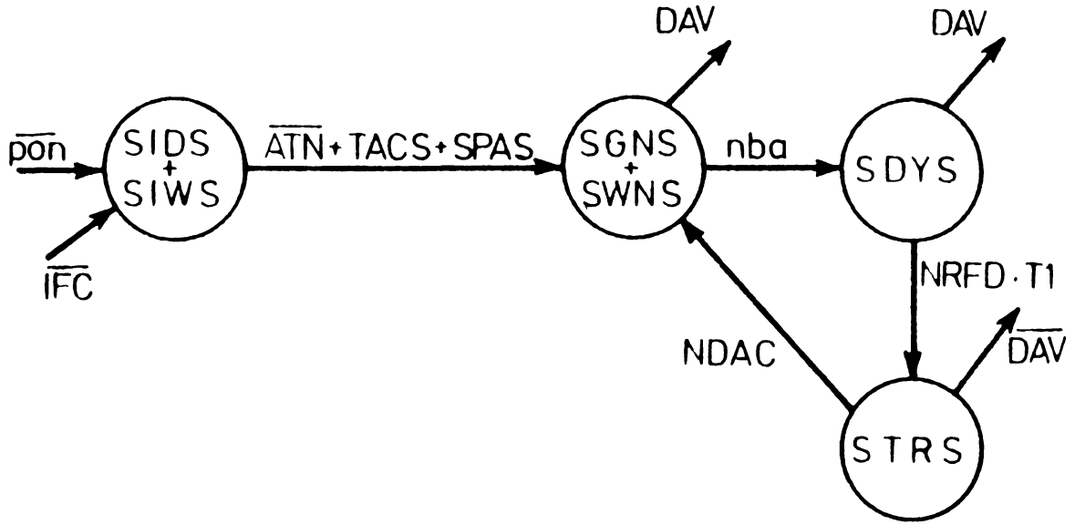


Fig.4.7.

Schema electrică corespunzătoare se prezintă în fig.4.8.

S-a obţinut o realizare mai simplă. Cu linie continuă s-a prezentat varianta sincronă, iar cu linie întreruptă elementele care pot înlocui circuitele bistabile sincrone, pentru a obţine o variantă asincronă.

In felul acesta se realizează oricare din cele lo funcţii. Modul lor de interconectare şi utilizare se prezintă pentru sistemul automat de testare. In acest caz, elementele de bază care trebuie urmărite, la realizarea interfeţelor cu fiecare modul, sînt legate de asigurarea unei fiabilităţi şi siguranţe în funcţionare ridicate. De aceste elemente se va ţine cont, în primul rînd, la stabilirea şi reţinerea capabilităţilor necesare pentru realizarea fiecărei funcţii a interfeţelor.

Blocul de comandă şi control al magistralei, BCCM, se prezintă în fig.4.9 şi conţine următoarele funcţii : C, T, L, AH,

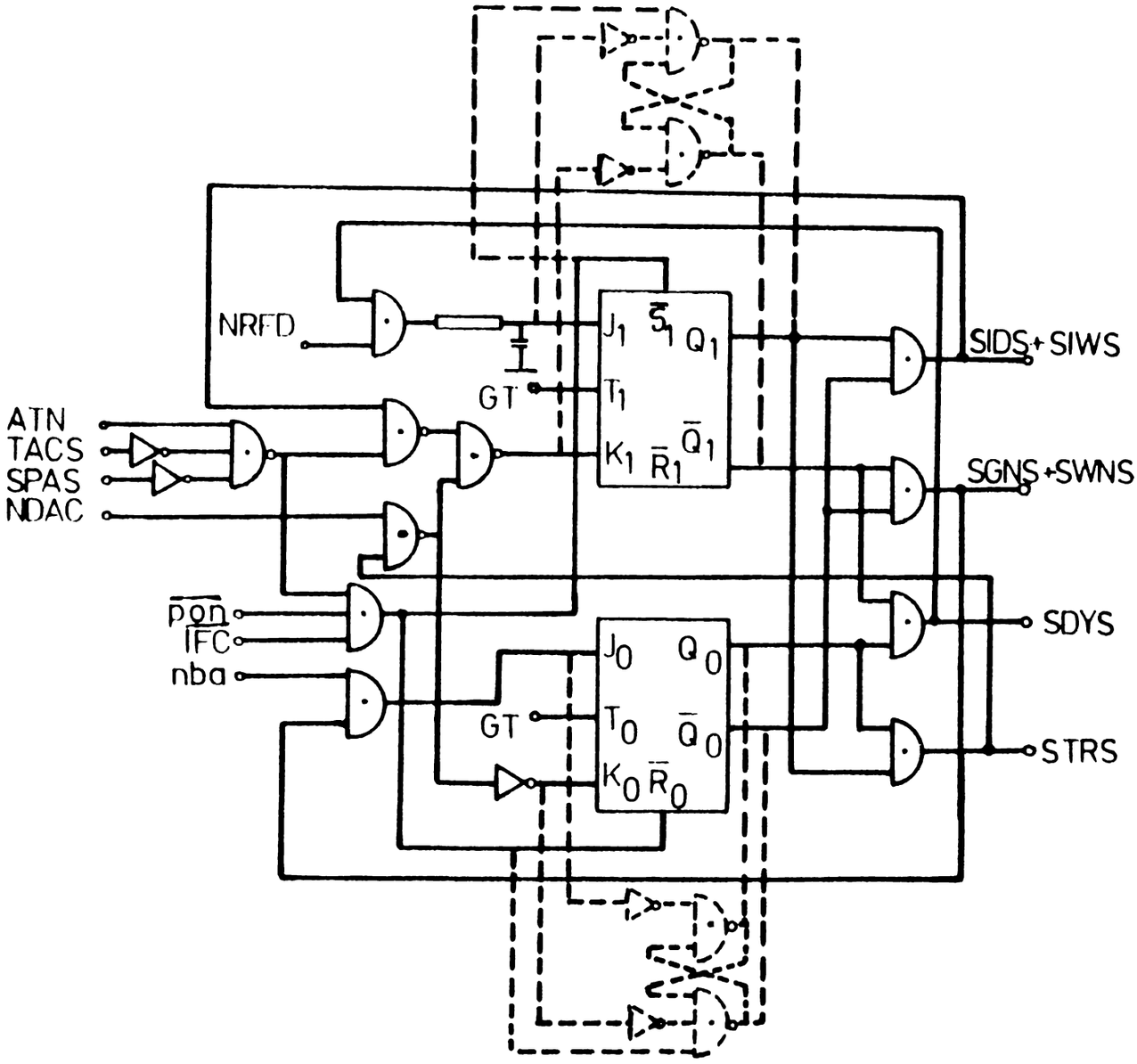


Fig.4.8.

SH, SR şi RL. Alături de acestea, în configuraţia blocului mai apar :

- registrul de mesaje locale, REMEL;
- registrul mesajelor de comandă, REMEC;
- decodificatorul mesajelor locale, DEMEL;
- decodificatorul mesajelor de comandă, DEMEC;
- registrul de recepţie a datelor, RERED;
- registrul de transmisie a datelor, RETAD;
- registrul cererilor de acces, RECA;
- decodificatorul de adrese, DADR;
- generatorul de comenzi pentru interfaţare cu echipamentul de calcul, GCIC;

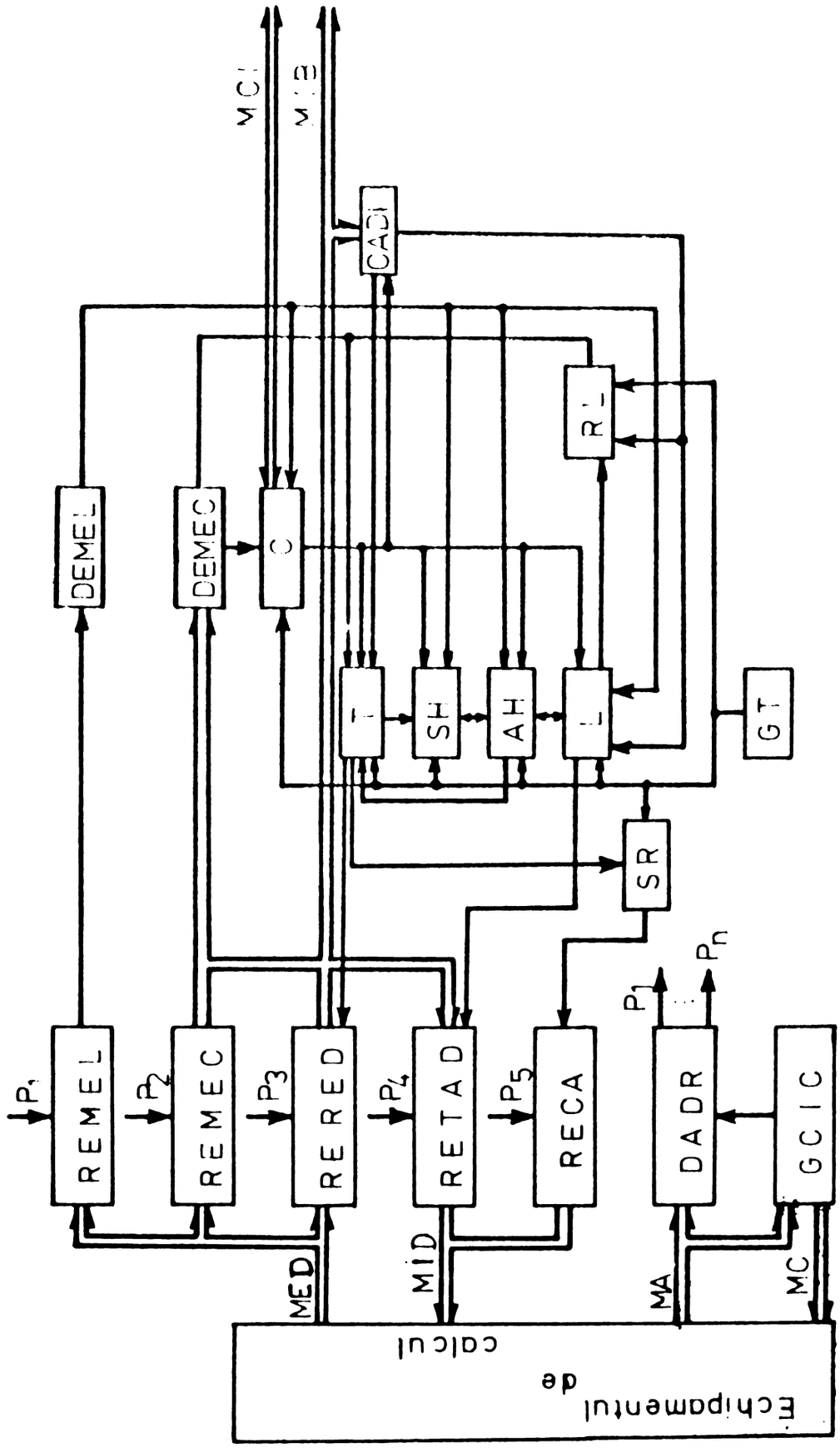

MC
MD
CADI
DEMEL
DEMEC
C
RL
I
SH
AH
L
GT
SR
P1
Pn
P1
P2
P3
P4
P5
REMEL
REMEC
RERED
RETAD
RECA
DADR
GCIC
MED
MID
MA
MC
calcul
de
Echipamentul

Fig.4.9.

- comparatorul de adrese, CADI;
- generatorul de tact al interfeţei, GT.

Adresele transmise pe magistrala MA sînt decodificate în blocul DADR, sub controlul semnalelor generate în blocul GCIC, obţinîndu-se astfel stabilirea porturilor Pi, utilizate pentru transferul informaţiei utile.

Mesajele de comandă, utilizate la adresarea modulelor, în stare nedecodificată, sînt analizate în blocul CADI. Adresa sau adresele fiecărui modul se cablează, iar în urma comparării cu mesajul emis de blocul BCCM, se ia decizia de activare sau nu a modulului respectiv.

Celelalte mesaje de comandă se decodifică în DEMEC, rezultînd comenzi pentru diferitele funcţii ale interfeţei.

Pe magistrala MIB se transmit atît mesajele de comandă destinate funcţiilor interfeţei, cît şi mesajele aferente modulelor. In această ultimă categorie intră : datele utile, adresele, comenzile necesare modulelor pentru a-şi realiza funcţiile, informaţiile de stare.

Funcţia RL nu este destinată exploatării curente, aceasta introducîndu-se numai pentru faza de punere în funcţiune, respectiv testare, cînd, dacă este activată, permite conectarea unui panou local.

Blocul BCCM asigură schimbul de informaţii cu echipamentul de calcul, iar cu ajutorul funcţiilor T, SH, L, AH şi SR realizează, prin magistrala MIB, transferul de informaţii cu celelalte module ale sistemului. In acest caz, respectivele module vor fi echipate, fiecare, fie cu funcţii emiţătoare, fie receptoare, fie cu ambele.

Analizînd modul de lucru al modulului de comandă şi al celui de testare, rezultă necesitatea asigurării următoarelor funcţii, de către interfaţa asociată :

- modulul de comandă, MC, cere funcţiile T, SH, L, AH, DC, DT, SR şi RL;
- modulul de testare, MT, cere, pentru aplicarea semnalelor pe pinii plachetei ce se testează, funcţiile : L, AH, DC, DT, ST şi RL, iar pentru înregistrarea stării pinilor, funcţiile : T, SH, AH, DC, SR şi RL.

Deoarece activităţile realizate de cele două module sînt distincte şi nu se suprapun, din punctul de vedere al accesului la magistrala MIB, se utilizează o singură interfaţă IRES pentru ambele module.

Funcţiile SR şi PP nu sînt absolut necesare amîndouă. Intr-un sistem de testare, care, în general, conţine un număr relativ redus de module, funcţia SR satisface solicitările practice, la un nivel acceptabil. Este indicată utilizarea ambelor funcţii în sistele mai complexe, ca număr şi tipuri de module (numerice, analogice şi/sau hibride). In acest caz este importantă obţinerea unei viteze de interogare sporită, ceea ce se realizează prin utilizarea combinată a funcţiilor SR şi PP.

Din punctul de vedere al modularităţii în cadrul sistemului de testare, partiţionarea interfeţei în funcţii fiind foarte bine făcută, este mai avantajoasă proiectarea unei interfeţe complete, cu caracter emiţător/receptor, iar orientarea ei spre un mod de lucru (de exemplu emiţător sau receptor) sau altul, să se facă în funcţie de modulul la care se asociază, printr-o echipare tehnologică potrivită. In felul acesta se asigură o interschimbabilitate rapidă a interfeţelor, reducerea timpului de punere în funcţiune, creşterea siguranţei în exploatare. In fig. 4.1o se prezintă schema bloc a interfeţei cu caracter emiţător/ receptor, IRES, utilizată în sistemul de testare.

De la blocul emiţător al unui modul, BE, se preiau date şi informaţii de stare, acest ultim tip, în faza de interogare serie sau paralelă. Tipul informaţiei se alege cu ajutorul multiplexorului MUX. Informaţiile utile sînt destinate blocurilor receptoare, BE, ale modulelor.

Pe magistrală, informaţia se transmite asincron, dar fiecare funcţie îşi desfăşoară activitatea proprie în mod sincron. Iniţiativa în activarea funcţiilor şi efectuarea schimbului o are echipamentul de calcul. După stabilirea funcţiei C active, se trece la emiterea mesajelor de comandă şi, în ultimă instanţă, informaţiile destinate modulelor. Acestea din urmă circulă între un modul cu funcţia T a interfeţei asociate, activă şi unul sau mai multe module cu funcţiile L ale interfeţelor asociate, active.

In concluzie, se poate afirma că acest tip de magistrală universală asigură un schimb de informaţii la o viteză mare de transfer (1 MHz, mult superioară magistralelor seriale), utilizează un număr mic de linii (în comparaţie cu magistralele S1OO, IEEE583), nu este orientată spre diferite tipuri de circuite (cum sînt cele ale microprocesoarelor) şi este foarte uşor de exploatat, în condiţii de siguranţă şi fiabilitate sporite.

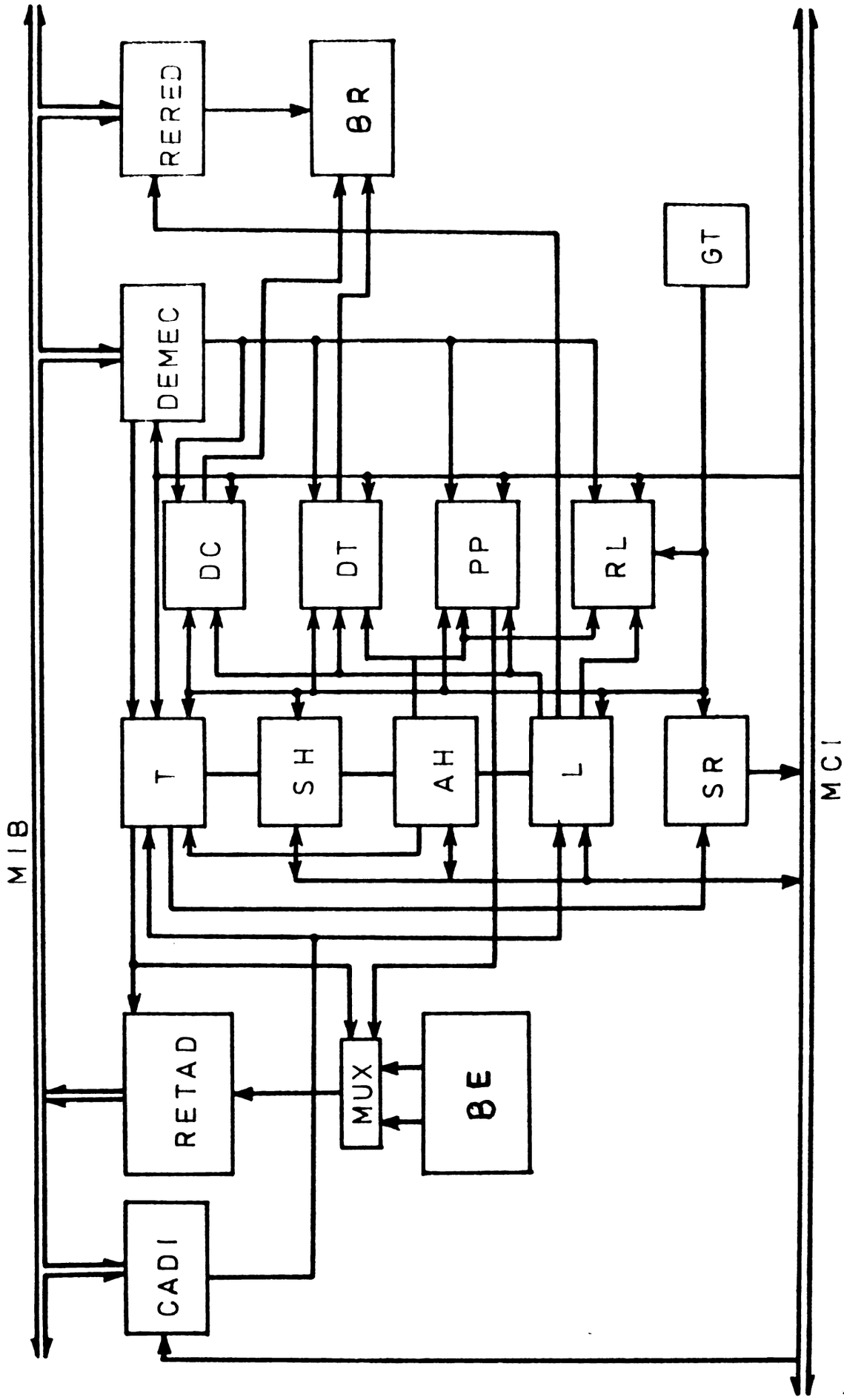

RERED
BR
GT
DEMEC
DC
DT
PP
RL
T
SH
AH
L
SR
MIB
MCI
RETAD
MUX
BE
CADI

Fig.4.1o.

4.4. Modulul de comandă al testării

Faza de testare presupune desfăşurarea a două etape distincte :

- etapa de pregătire a operaţiei de testare;
- testarea propriu-zisă.

In prima etapă se stabileşte modul de lucru, tipul testării ce se va efectua, tipul pinilor şi al stimulilor ce se aplică pe fiecare pin în parte. In etapa a doua se comandă aplicarea simultană şi sincronă a semnalelor asupra plachetei care se testează. Citirea răspunsurilor se face, de asemenea controlat, în modulul de testare.

Schema bloc a modulului se prezintă în fig.4.11. Semnificaţia notaţiilor utilizate este următoarea :

- amplificatorul de magistrală, AM;
- distribuitorul de informaţii şi comenzi, DISIC;
- registrul adresei iniţiale, RAI;
- registrul adresei finale, RAF;
- numărătoarele de adrese ale blocurilor de memorie, NADR1 şi NADR2;
- comparatorul adresei finale, CADR;
- decodificatorul modurilor de lucru, DML;
- numărătorul de scriere/citire, NCS;
- decodificatorul de scriere/citire, DSC;
- numărătorul condiţiei de oprire, NCOP;
- comparatorul condiţiei de oprire, CCOP;
- registrul condiţiei de oprire, RECOP;
- multiplexorul condiţiei de oprire, MUCOP;
- numărătorul de tacte, NT;
- numărătorul de întîrzieri, NI;
- multiplexorul numărului de tacte, MUNT;
- generatorul de comenzi sincrone, GCS.

Registrele RAI şi RAF se utilizează pentru păstrarea adresei iniţiale, respectiv finale, necesare blocurilor MVT şi MRVT, care se utilizează pentru testarea la viteză mare. Modificarea adresei între valoarea iniţială şi finală se realizează cu ajutorul numărătoarelor NADR1, respectiv NADR2.

Blocul CADR realizează o comparare între adresa finală şi adresa curentă. In caz de egalitate, dacă modul de lucru este cel fără ciclare, se blochează numărarea şi prin aceasta, întreaga eta-

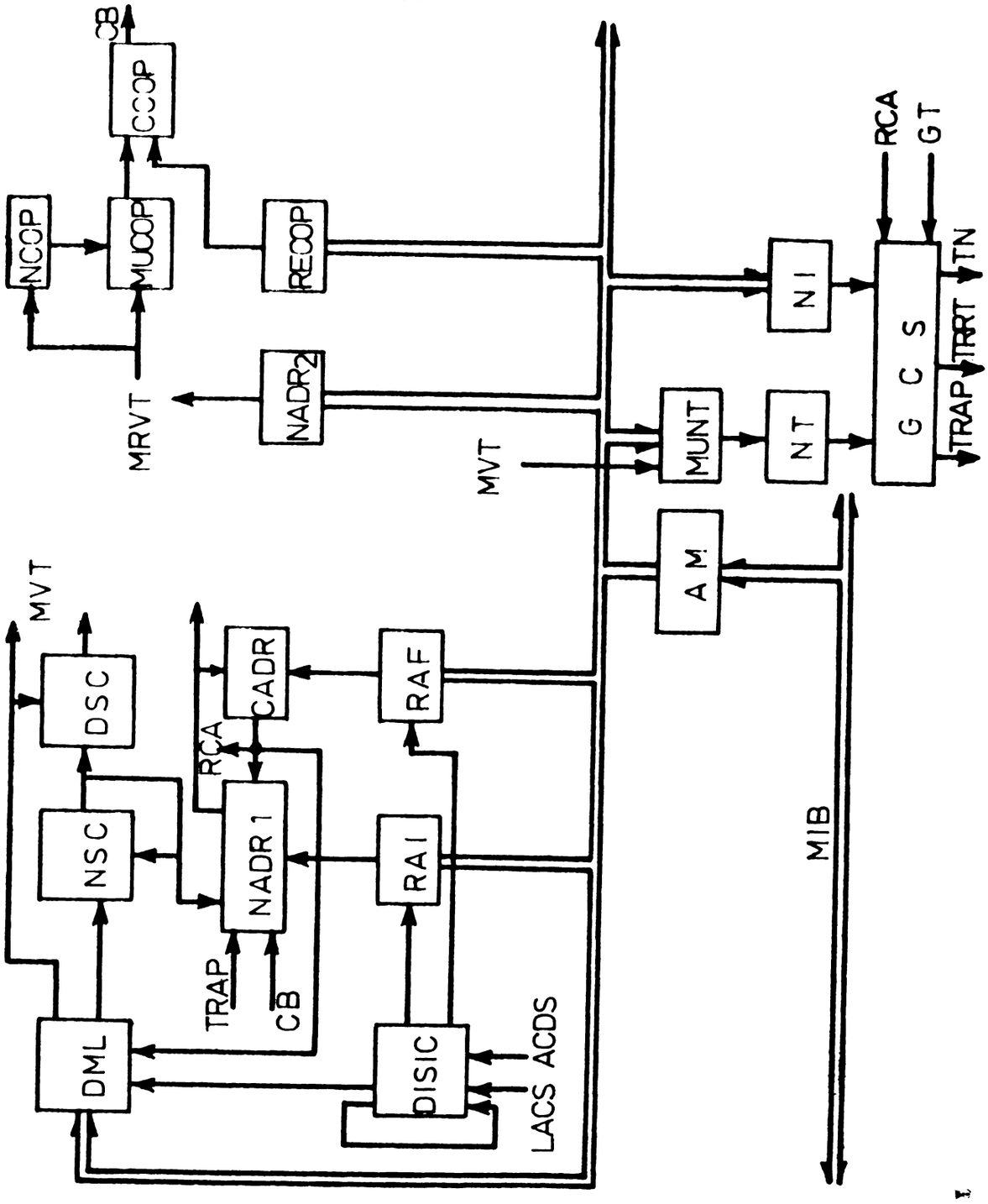


Fig.4.11.

pă de testare propriu-zisă. Dacă modul de lucru este cel cu ciclare, reluarea se face, automat, de la adresa inițială.

Blocul DML stabilește modurile de lucru, în cazul testării cu ajutorul blocurilor MVT , MRVT și anume : încărcare MVT, citire MRVT, testarea cu memoria rapidă (MVT, MRVT). Totodată, împreună cu blocurile NSC și DSC, asigură selecția memoriei și accesul la nivel de octet. Incărcarea vectorilor de testare, în MVT, extragerea lor pentru a fi aplicați asupra plachetei și înregistrarea rezultatelor obținute, în MRVT, se poate realiza la

viteza maximă de lucru a memoriei utilizate, programînd adecvat frecvenţa tactului numărătoarelor NSC, respectiv NADR1 şi NADR2.

Amplificatorul de magistrală, AM, s-a introdus cu scopul de a asigura încărcarea magistralei MIB cu o singură sarcină TTL, în situaţiile în care mai multe blocuri ale unui modul utilizează informaţii de pe magistrală.

Blocul DISIC are rolul de a stabili blocurile care participă la efectuarea unei operaţii de testare şi de a genera semnalele de comandă necesare fiecărui bloc activat. Este solicitat de multitudinea şi complexitatea blocurilor, respectiv a modulelor, care concură la aceeaşi interfaţă IRES. Acest bloc este un automat secvenţial format, în principal, dintr-un numărător,NRS şi un decodificator, DS. O anumită stare a numărătorului specifică o operaţie dată, pentru care se activează anumite blocuri. In repaus, acesta se află în starea iniţială, menţinută de funcţia L, prin starea sa de repaus, $\overline{\text{IACS}}$. Activarea altei operaţii se poate face numai în urma trecerii prin starea iniţială. Se apreciază că acest mod de lucru oferă o siguranţă sporită în funcţionare. Specificarea unei anumite operaţii se face prin introducerea în numărător a unei informaţii prestabilite,de pe magistrală, iar comenzile aferente rezultă în urma decodificării stărilor prin care trece numărătorul, ce-şi modifică conţinutul la fiecare apel la interfaţă. Ultima comandă aferentă operaţiei de testare activă blochează numărătorul pe starea curentă. Diagrama de tranziţii a blocului se prezintă în fig.4.12.

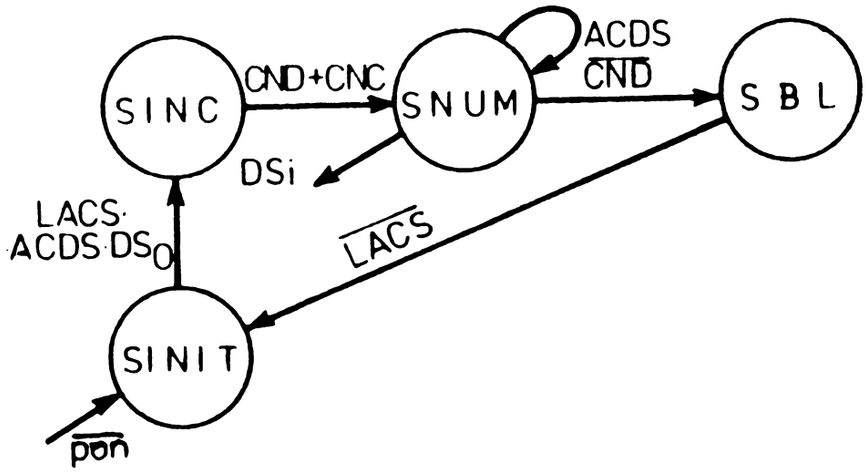


Fig.4.12.

Semnificaţia notaţiilor utilizate este următoarea :

- starea iniţială, SINIT;
- starea de încărcare, SINC;
- starea de numărare, SNUM;

- starea de blocare, SBL;
- stările de numărare decodificate, $DS_{i,i} = o,11$;
- comanda de numărare în sens descrescător, CND;
- comanda de numărare în sens crescător, CNC;
- starea de acceptare a unui mesaj, ACDS, caracteristică funcţiei AH.

În etapa de pregătire a testării se realizează următoarele operaţii :

a) - stabilirea tipului pinilor : de intrare sau de ieşire. Aceasta se realizează prin înscrierea informaţiei corespunzătoare în blocul SIE;

b) - stabilirea familiei logice în care se încadrează funcţionarea fiecărui pin, acţiune ce se realizează prin înscrierea informaţiei corespunzătoare în blocul SFL;

c) - distribuirea stimulilor de tip nivel logic pe fiecare pin, ceea ce se realizează cu ajutorul blocului DNL;

d) - distribuirea stimulilor aleatori se realizează cu ajutorul blocului DSA.

Pentru efectuarea acestor operaţii se solicită transmisia, într-o succesiune ciclică, de mesaje utile, care să conţină informaţiile :

- adresa pinului;
- informaţia necesară pinului respectiv;

până la epuizarea numărului de pini ai plachetei ce se testează. Modul de lucru se prezintă în schema logică din fig.4.13.

e) - distribuirea stimulilor de tip tacte. Această acţiune presupune transmiterea următoarelor informaţii :

- adresa pinului;
- informaţia necesară pinului respectiv;
- numărul de tacte necesare pe pinul respectiv;
- polaritatea tactelor;
- numărul de întîrzieri.

Operaţia se realizează cu ajutorul blocului DT, iar informaţiile se pot transmite în aceeaşi succesiune ca în cazurile anterioare, pentru numărul de tacte sau de întîrzieri, transmiţîndu-se, mai întîi, o adresă a numărătorului de tacte, NT, respectiv de întîrzieri, NI, după care urmează numărul propriu-zis. Bitul de pe poziţia cea mai semnificativă a numărătorului NT va indica polaritatea tactelor. Deci este valabilă, şi în acest caz, schema logică din fig.4.13.

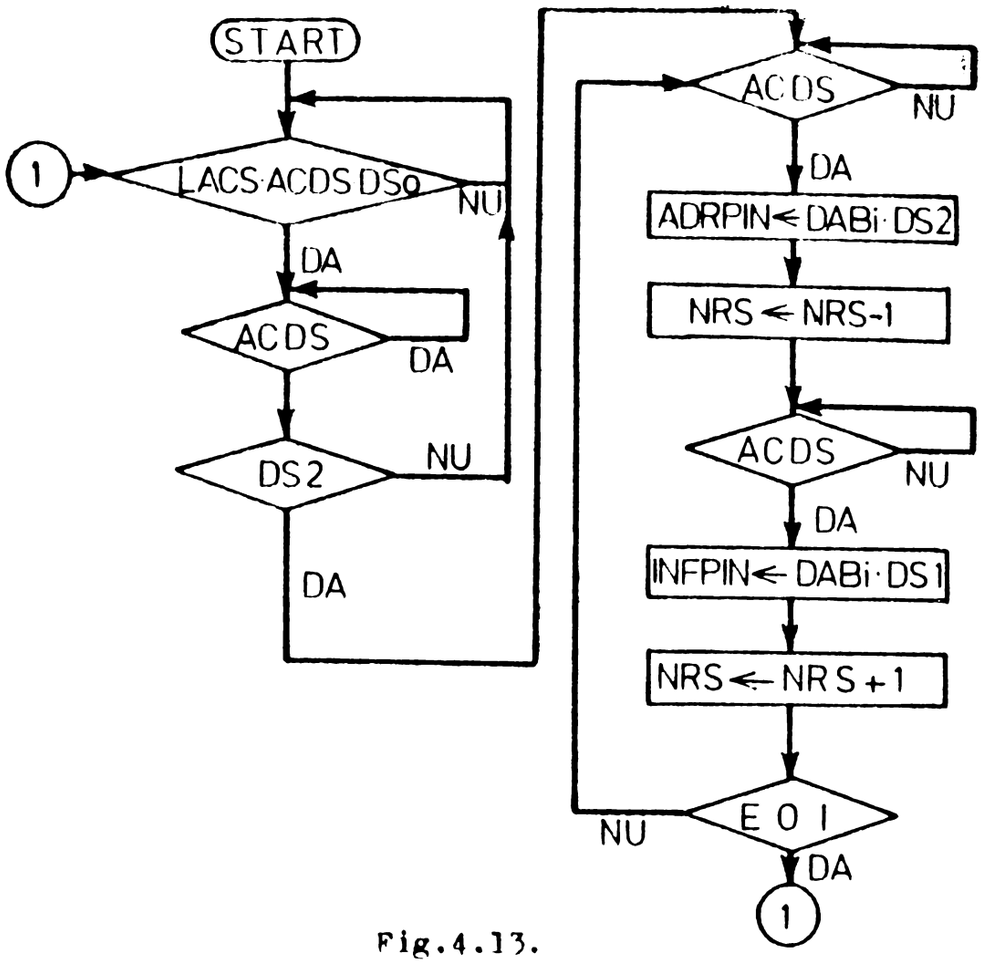


Fig.4.13.

f) - testarea cu ajutorul blocurilor MVT şi MRVT. In acest caz se transmit următoarele informaţii :

- adresa iniţială a memoriei, ADRI;
- adresa finală a memoriei, ADRF;
- condiţia de oprire, COP;
- modul de lucru (MD) - încărcare MVT;
  - testare cu blocurile MVT şi MRVT;
  - citirea MRVT în echipamentul de calcul.

Succesiunea acţiunilor caracteristice acestei operaţii se prezintă în fig.4.14.

Oprirea fazei de testare propriu-zisă pe o anumită condiţie îndeplinită se realizează cu ajutorul blocurilor : NCOP, care specifică condiţia de blocare sub formă de număr de tranziţii îndeplinit, RECOP, ce conţine condiţia prestabilită pentru oprire, MUCOP, prin care se realizează multiplexarea tipului de informaţie din blocurile MVT sau NCOP, ce urmează a fi comparată cu cea existentă în registrul RECOP, acţiune realizată cu ajutorul blocului CCOP. In caz de egalitate se blochează activitatea numărătoarelor de adresă şi deci, operaţia de testare.

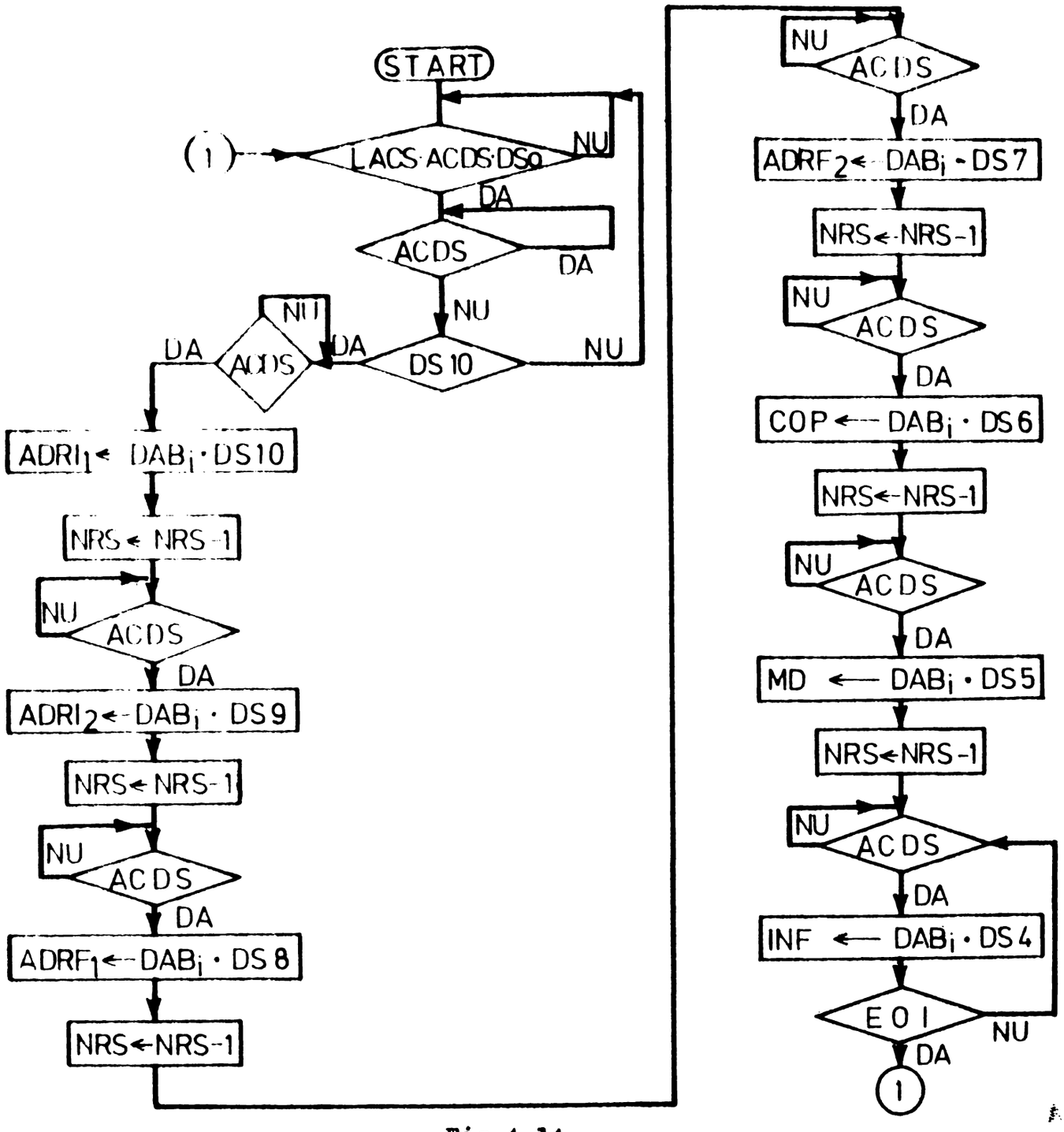


Fig.4.14.

Numărătorul NT se utilizează pentru testarea schemelor care conţin, în configuraţia lor, şi circuite secvenţiale. Acesta permite aplicarea, pe un anumit pin, aferent unui vector de testare, un număr prestabilit de tacte. Numărătorul NT se poate încărca din blocul MVT sau de pe magistrala MIB, utilizînd multiplexorul MUNT.

Numărătorul NI este destinat generării de întîrzieri programate, necesare, în special, testării circuitelor asincrone.

Blocul OCS este destinat generării semnalelor de comandă necesare în faza de testare propriu-zisă. Acestea sînt :

- TRAP - semnalul care precizează momentul aplicării semnalelor stabilite prin vectorul de testare, asupra plachetei care se verifică;
- TRRT - semnalul care stabileşte momentul înregistrării răspunsului plachetei, obţinut în urma aplicării semnalelor specificate prin vectorul de testare;
- TN - semnale de tact, necesare funcţionării circuitelor secvenţiale de pe placheta ce se testează.

Faza de testare propriu-zisă constă în aplicarea sincronă, a semnalelor specificate prin vectorii de testare, asupra plachetei şi înregistrarea controlată a răspunsurilor. După stabilizarea semnalelor pe intrările de date ale circuitelor combinaţionale şi secvenţiale se generează, sincron, numărul de tacte necesar. La epuizarea acestuia, odată cu apariţia semnalului TRRT se înregistrează răspunsul plachetei. In cazul inexistenţei circuitelor secvenţiale, semnalul TRRT se generează, sincron cu tactul, imediat după semnalul TRAP. Modul de generare al comenzilor se prezintă în fig.4.15.

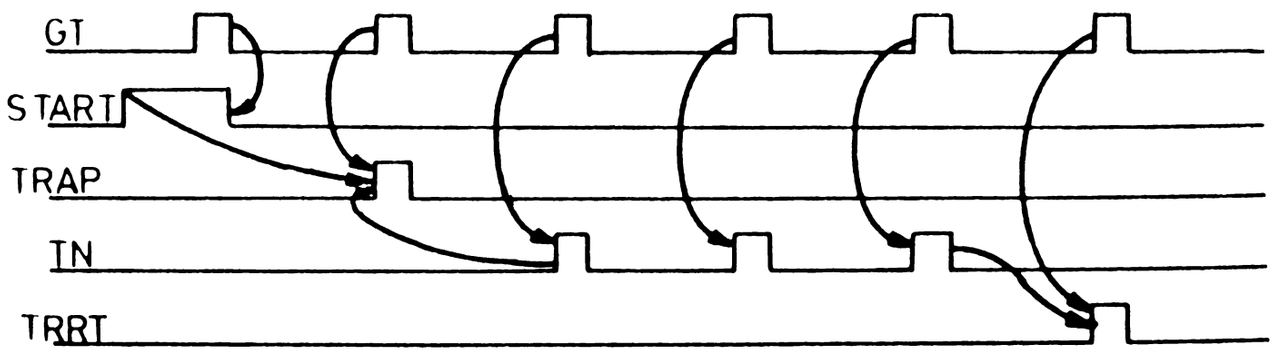


Fig.4.15.

Succesiunea operaţiilor în faza de testare se prezintă în schema logică din fig.4.16.

4.5. Modulul de testare

Cu ajutorul acestui modul se realizează următoarele :

- stabilirea tipului pinilor de intrare sau de ieşire;
- stabilirea familiei logice în care se încadrează funcţionarea fiecărui pin;

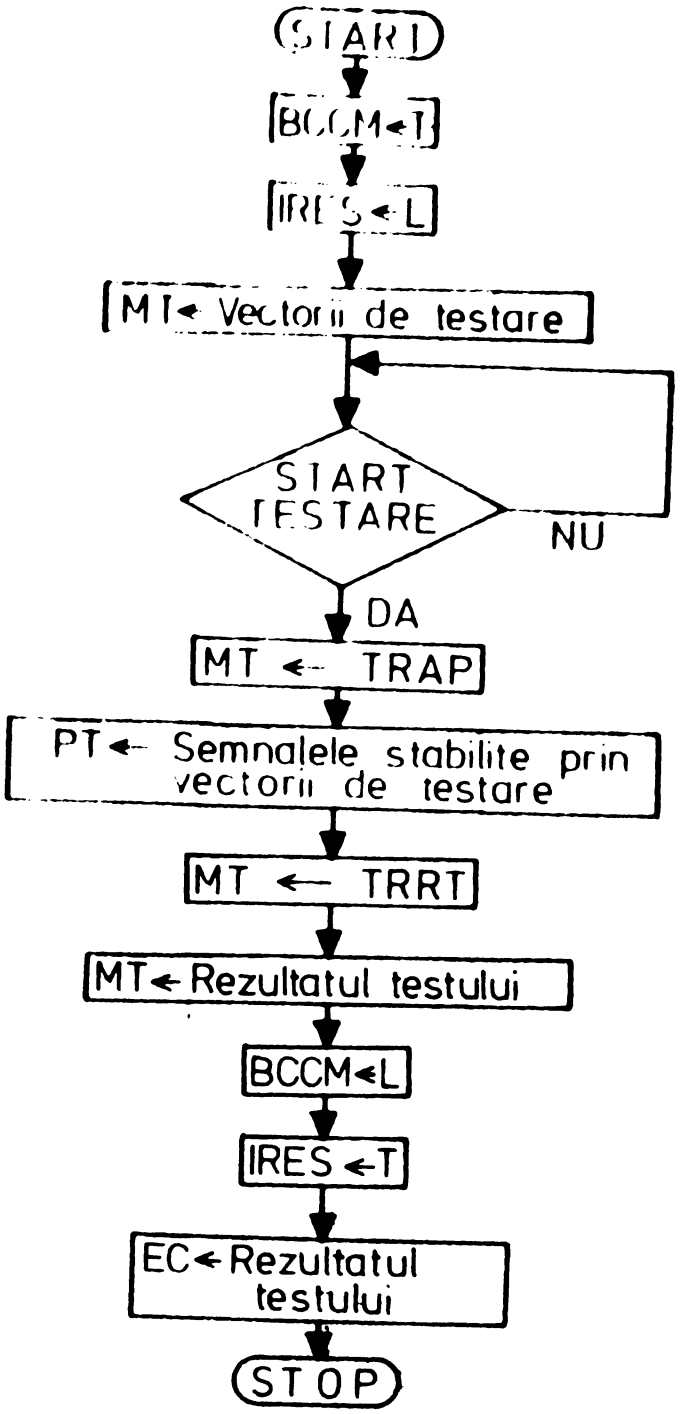


Fig.4.16.

distribuirea stimulilor de tip nivel logic pe fiecare pin;

distribuirea stimulilor aleatori pe fiecare pin;

- distribuirea stimulilor de tip tacte, pe fiecare pin;
- înregistrarea răspunsurilor plachetei care se testează;
- înregistrarea unui număr dat de vectori de testare într-o memorie intermediară, cu scopul de a-i aplica, ulterior, asupra plachetei, în vederea obținerii unei viteze sporite de testare. Inregistrarea răspunsurilor, în acest caz, se face într-o altă memorie intermediară. Viteza de testare este limitată superior numai de timpul de acces al memoriei utilizate.

Configuratia modulului se prezintă în fig.4.17. În esență,

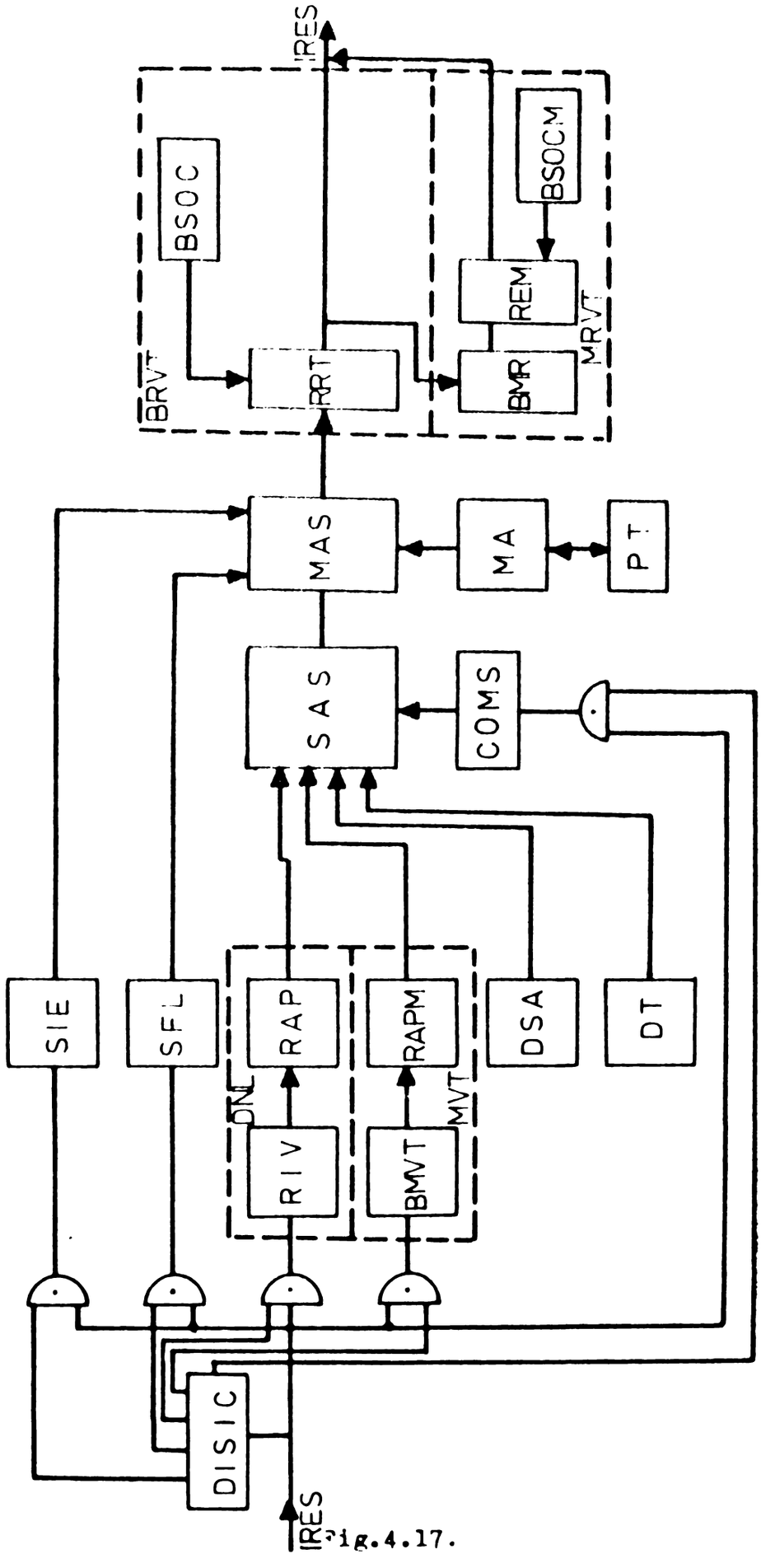


Fig.4.17.

elementul principal al blocurilor SIE, SFL, DNL, DSA, DT, BRVT îl constituie cîte un registru a cărui lungime activă este identică cu numărul de pini ai plachetei ce se testează, fiecărui pin corespunzîndu-i o poziţie în registru. Această poziţie, adresată, va putea memora informaţia necesară stabilirii tipului pinului sau caracterul stimulului ce se va aplica pe acesta.

In faza de pregătire a testării se stabileşte tipul pinilor şi al informaţiei ce urmează a fi aplicată pe un anumit pin, utilizînd blocurile SIE, SFL, DT şi DSA. Vectorii de testare se încarcă în blocul DNL (în registrul de împachetare a vectorilor, RIV) sau în blocul MVT. Informaţia se extrage,din echipamentul de calcul,în unităţi de cîte 8 biţi, echivalente a 8 pini. In faza de testare propriu-zisă se comandă transferul sincron al informaţiei din registrul RIV în registrul RAP, respectiv din blocul de memorie BMVT în registrul RAPM, în funcţie de modul de lucru, asigurîndu-se astfel aplicarea simultană a semnalelor asupra tuturor pinilor de intrare ai plachetei.

Blocul SAS conţine un circuit de multiplexare cu 8 căi, care, împreună cu registrul de selecţie COMS, realizează selectarea atributelor semnalelor ce se aplică pe un anumit pin.

Pentru înregistrarea răspunsurilor plachetei ce se verifică se utilizează următoarele blocuri :

- registrul ce păstrează răspunsul obţinut în urma aplicării semnalelor specificate de un singur vector de testare asupra plachetei, RRT;
- blocul de selectare a informaţiei ce urmează a fi introdusă în echipamentul de calcul, BSOC;
- blocul de memorie pentru înregistrarea răspunsurilor obţinute în cazul aplicării semnalelor specificate de un număr de vectori de testare, MRVT. Răspunsurile se introduc în memoría BMR, printr-un registru tampon, la care înscrierea este comandată prin semnalul TRRT. Deoarece registrul are un timp de acces mult inferior memoriei, se asigură fixarea precisă a momentului de apariţie al răspunsului. Răspunsul sosit după dispariţia semnalului TRRT nu mai este înregistrat, ceea ce este un indiciu că timpul de propagare prin plachetă este mai mare decît cel dorit, deci, din acest punct de vedere, placheta se poate considera defectă;
- registrul de ieşire al blocului de memorie, REM - conţine răspunsul care urmează a fi introdus în echipamentul de calcul;

- blocul de selectare a informaţiei ce se citeşte din registrul REM, BSCCM.

Pentru introducerea, în echipamentul de calcul, a rezultatului testării, se activează funcţia T a interfeţei IRES şi funcţia L a blocului BCCM, citirea efectuîndu-se, pentru întregul răspuns, echivalent în lungime, cu numărul pinilor, din registrul RRT.

Răspunsurile obţinute în urma aplicării unui număr dat de vectori de testare se citesc din blocul MRVT, prin intermediul registrului REM.

Funcţiei T îi sînt afectate două adrese, una activează citirea din registrul RRT, iar cealaltă din registrul REM.

4.6. Modulul de atac/sesizare

In realizarea practică a modulului de atac/sesizare se cristalizează două orientări principale şi anume : realizare cu relee şi realizare cu dispozitive electronice.

Prima variantă este întîlnită mai ales la sistemele care asigură testarea şi a plachetelor echipate cu circuite analogice. In acest caz se realizează o separare galvanică foarte bună, cădere mică de tensiune pe contactele releelor, dar la o viteză de comutaţie scăzută.

Pentru sistemele destinate testării plachetelor echipate cu circuite logice se utilizează, tot mai des, al doilea mod de construcţie, care asigură, în principal, o viteză de comutaţie net superioară.

Acest modul are, în configuraţia sa, două circuite de bază: circuitul de atac (A) şi circuitul de sesizare (S).

Circuitul A realizează aplicarea informaţiei pe pin, la nivelul energetic cerut de acesta. Pentru a-şi putea realiza funcţia, circuitul A trebuie să prezinte trei stări : o stare pasivă, în care să nu influenţeze valoarea informaţiei de pe pin, o stare activă cu nivelul logic "1" şi o stare activă cu nivelul logic "0". Datorită similitudinii funcţionale, circuitul A poate fi realizat într-o configuraţie asemănătoare cu etajul de ieşire în contratimp a porţilor familiei TTL. Cîteva modificări ce apar sînt cauzate de necesitatea asigurării unei puteri mai mari decît în cazul porţilor TTL şi a unei protecţii la scurtcircuit.

Circuitul S stabileşte starea fiecărui pin, semnalizînd totodată şi corectitudinea nivelelor logice (0-0,4 V pentru ni-

velul logic "0" corect şi 2,4-5 V pentru nivelul logic "1" corect, obţinute la ieşirea unui circuit al familiei TTL).

Schema echivalentă a modulului de atac/sesizare se prezintă în fig.4.18, în care se reţine şi influenţa pinului plachetei ce se testează, prin R_P şi E_P. Condiţiile idealizate de funcţionare sînt :

- pentru S : $R_S \to \infty$, $E_S \approx 0$;
- pentru A în stare pasivă : $R_D \to \infty$, $E_D \approx 0$;
- pentru A în stare activă : $R_D \approx 0$, iar E_D avînd valoarea corespunzătoare nivelului logic ce se aplică pe pin.

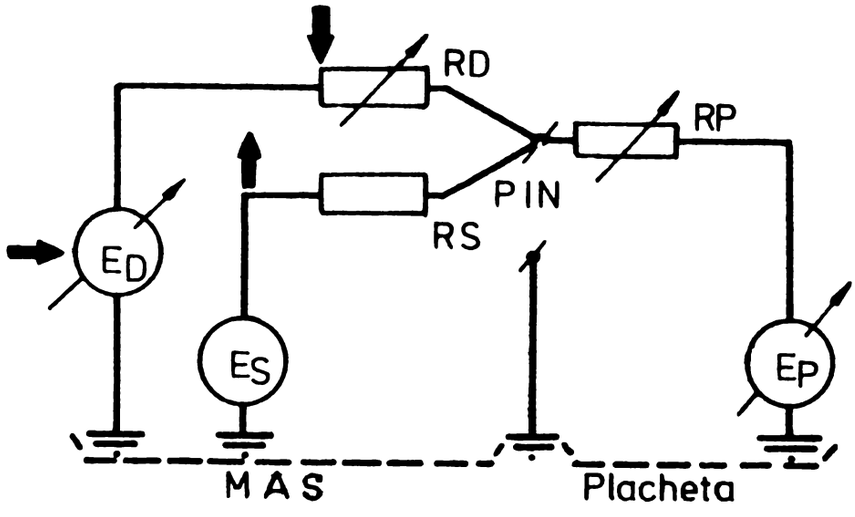


Fig.4.18.

Parametrii reali ai schemei echivalente din fig.4.18 ţin cont de o serie de considerente practice. Astfel, rezistenţa R_S se alege cu o astfel de valoare încît să se poată sesiza întreruperea legăturii spre sau de la plachetă.

Pentru circuitul A interesează valoarea rezistenţei R_D, pentru starea pasivă şi curentul şi tensiunea furnizată în stare activă. Caracteristica statică a circuitului A, în stare activă, pentru nivelele logice "1" şi "0", se prezintă în fig.4.19.

La apariţia unui scurtcircuit pe plachetă, între un pin şi masă, curentul debitat de circuitul A rămîne limitat la valoarea nedistructivă I_{HS}. Analog, în starea activă "0", circuitul menţine o tensiune între 0 şi U_{LO}, pentru un curent nominal maxim absorbit de valoare I_{LO}. La scurtcircuit pe plachetă, în situaţia cea mai dificilă, curentul absorbit este limitat la valoarea nedistructivă I_{LS}.

Timpii de comutaţie ai modulului nu trebuie să depăşească 10 nS.

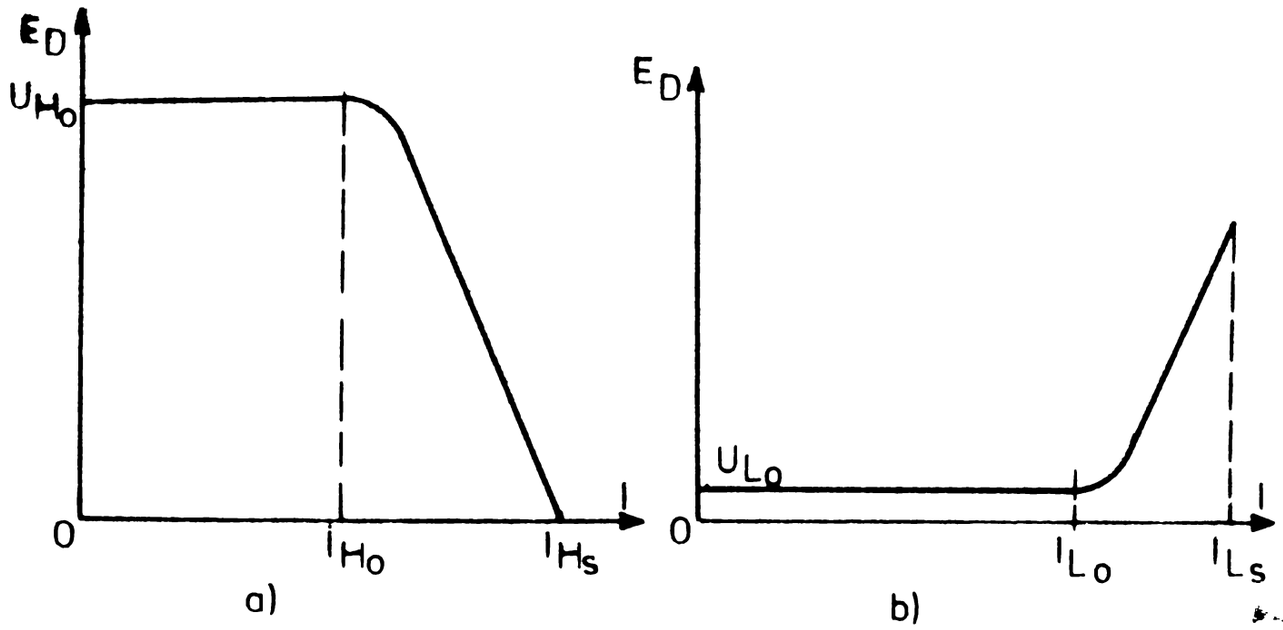


Fig.4.19.

Pentru a obţine performanţele prezentate, modulul de atac/sesizare se realizează cu semiconductoare discrete, circuitele logice integrate preluînd funcţiile de comandă şi de interpretare a rezultatelor.

4.7. Posibilităţi de extindere ale sistemului automat de testare

Fiind realizat modular, sistemului automat de testare i se pot adăuga blocuri noi, care să-i confere noi capabilităţi în testarea plachetelor echipate cu circuite logice SSI, MSI şi LSI. Cîteva din aceste blocuri sînt următoarele :

a) - Sonda logică - permite înregistrarea stării logice a unui nod de pe placheta echipată. Din punctul de vedere al sistemului, sonda este considerată ca un pin de ieşire mobil, al plachetei. Stabilirea nodului a cărui stare logică va fi citită, la un moment dat, se poate face automat, prin program, cînd sonda este ghidată spre nodul în cauză sau manual, situaţie în care utilizatorul este cel care stabileşte nodul a cărui stare logică urmează a fi citită. In felul acesta se urmăreşte obţinerea unor rezoluţii cît mai bune, în faza de localizare a defectelor. Utilizarea, în această fază, a sondei logice, împreună cu un simulator de circuite, poate oferi noi avantaje substanţiale, legate de localizarea rapidă a defectelor.

b) - Blocul "In Circuit" - utilizînd un adaptor cu pat de cuie,

realizează accesul direct pe placheta care se verifică. In felul acesta cresc foarte mult posibilităţile de localizare a defectelor. Totodată permite efectuarea şi a unei testări funcţionale şi/sau parametrice a circuitelor, pentru care s-a asigurat accesul prin pat de cuie. Programele de testare devin mult mai simple, iar realizarea lor cu ajutorul sistemului prezentat în cap.3 nu ridică nici o problemă deosebită. Singurul element dificil de realizat rămîne patul de cuie.

c) - Blocul de emulare - destinat punerii în funcţiune a sistemelor realizate cu circuite LSI - configuraţia a fost prezentată în cap.2.

4.8. Exemplu de testare a plachetelor cu ajutorul sistemului automat de testare

Pentru verificarea unei scheme este necesar să se cunoască vectorii de testare care activează fiecare cale ce există între intrările şi ieşirile schemei , precum şi valorile logice ale ieşirilor cînd, pe intrări, se aplică respectivii vectori. Determinarea setului complet al vectorilor de testare şi a valorii logice a ieşirilor respective, pentru o schemă dată, se prezintă cu ajutorul exemplului din fig.4.2o.

a) Analiza ieşirii Z_1.

$$Z_1 = X_1X_2\bar{X}_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2X_3\bar{X}_4\bar{X}_5 + X_1X_2\bar{X}_3X_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1\bar{X}_2\bar{X}_3X_4\bar{X}_5 + X_1X_2X_3X_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1X_2X_3\bar{X}_4X_5 ;$$

Intrările / Vectorii de testare	X_1	X_2	X_3	X_4	X_5	Ieşirea Z_1	Calea activată
VT_1	P	1	0	0	0	P	$X_1 - Y_4 - Z_1$
VT_2	1	P	0	0	0	P	$X_2 - Y_1 - Y_3 - Y_4 - Z_1$
VT_3	1	0	P	0	0	P	$X_3 - Y_1 - Y_3 - Y_4 - Z_1$
VT_4	1	0	0	P	0	P	$X_4 - Y_2 - Y_3 - Y_4 - Z_1$
VT_5	1	0	0	0	P	P	$X_5 - Y_2 - Y_3 - Y_4 - Z_1$

b) Analiza ieşirii Z_2.

$$Z_2 = X_2\bar{X}_3\bar{X}_4\bar{X}_5X_6\bar{X}_7 + \bar{X}_2X_3\bar{X}_4\bar{X}_5X_6\bar{X}_7 + X_2\bar{X}_3X_4X_5\bar{X}_6\bar{X}_7 +$$
$$\bar{X}_2X_3\bar{X}_4X_5\bar{X}_6\bar{X}_7 + \bar{X}_2\bar{X}_3X_4\bar{X}_5X_6\bar{X}_7 + X_2X_3\bar{X}_4\bar{X}_5\bar{X}_6\bar{X}_7 +$$
$$\bar{X}_2\bar{X}_3X_4X_5\bar{X}_6\bar{X}_7 + X_2\bar{X}_3X_4X_5X_6\bar{X}_7 ;$$

Intrările / Vectorii de testare	X_2	X_3	X_4	X_5	X_6	X_7	Ieșirea Z_2	Calea activată
VT_1	P	0	0	0	1	0	P	$X_2-Y_1-Y_3-Y_6-Z_2$
VT_2	0	P	0	0	1	0	P	$X_3-Y_1-Y_3-Y_6-Z_2$
VT_3	0	0	P	0	1	0	P	$X_4-Y_2-Y_3-Y_6-Z_2$
VT_4	0	0	0	P	1	0	P	$X_5-Y_2-Y_3-Y_6-Z_2$
VT_5	1	0	0	0	P	0	P	$X_6-Y_6-Z_2$
VT_6	1	0	0	0	1	P	$\bar{P}$	$X_7-Y_5-Y_6-Z_2$

e) Analiza ieșirii Z_3.

$$Z_3 = \bar{X}_1\bar{X}_6X_7X_8\bar{X}_9 + X_1\bar{X}_6\bar{X}_9Y_1^* + \bar{X}_6\bar{X}_8\bar{X}_9Y_1^* + X_6X_{10} ;$$

Intrările / Vectorii de testare	X_1	X_6	X_7	X_8	X_9	X_{10}	Y_1^*	Ieșirea Z_3	Calea activată
VT_1	0	0	P	1	0	0	X0	P	$X_7-Y_7-Y_9-Y_{10}-Y_{11}-Z_3$
VT_2	$\bar{P}$	0	1	1	0	0	X0	P	$X_1-Y_{12}-Y_{13}-Y_7-Y_9-Y_{10}-Y_{11}-Z_3$
VT_3	0	P	1	1	0	0	1	$\bar{P}$	$X_6-Y_8-Y_9-Y_{10}-Y_{11}-Z_3$
VT_4	0	P	0	1	0	1	0	P	$X_6-Y_{15}-Y_{10}-Y_{11}-Z_3$
VT_5	0	0	$\bar{P}$	1	0	0	X1	$\bar{P}$	$X_7-Y_7-Y_9-Y_{10}-Y_{11}-Z_3$
VT_6	0	0	1	P	0	0	X0	P	$X_8-Y_{13}-Y_7-Y_9-Y_{10}-Y_{11}-Z_3$
VT_7	0	0	1	1	P	0	X1	$\bar{P}$	$X_9-Y_{14}-Y_7-Y_9-Y_{10}-Y_{11}-Z_3$
VT_8	0	0	0	0	0	0	P	P	$Y_7-Y_9-Y_{10}-Y_{11}-Z_3$

d) Analiza ieșirii Z_4.

$$Z_4 = X_8\bar{X}_9X_{10}X_{11}X_{12} + \bar{X}_8\bar{X}_9Y_2^*Y_3^* ;$$

Intrările / Vectorii de testare	X_8	X_9	X_{10}	X_{11}	X_{12}	Y_2^*	Y_3^*	Ieşirea Z_4	Calea activată
VT_1	1	0	P	1	1	X0	1	P	X_{10}-Y_{16}-Y_{17}-Y_{18}-Z_4
VT_2	1	0	1	P	1	X0	1	P	X_{11}-Y_{16}-Y_{17}-Y_{18}-Z_4
VT_3	1	0	1	1	P	1	X0	P	X_{12}-Y_{19}-Y_{20}-Y_{18}-Z_4
VT_4	P	0	1	1	1	X0	1	P	X_8-Y_{17}-Y_{18}-Z_4
VT_5	P	0	1	1	1	1	X0	P	X_8-Y_{20}-Y_{18}-Z_4
VT_6	1	0	1	P	1	1	X0	P	X_{11}-Y_{16}-Y_{19}-Y_{20}-Y_{18}-Z_4
VT_7	1	0	$\bar{P}$	1	1	X1	X0	$\bar{P}$	X_{10}-Y_{16}-Y_{17}-Y_{18}-Z_4
VT_8	0	P	0	0	0	X0	X1	$\bar{P}$	X_9-Y_{14}-Y_{20}-Y_{18}-Z_4
VT_9	0	P	0	0	0	X1	X0	$\bar{P}$	X_9-Y_{14}-Y_{17}-Y_{18}-Z_4
VT_{10}	1	0	1	1	$\bar{P}$	X0	X1	$\bar{P}$	X_{12}-Y_{19}-Y_{20}-Y_{18}-Z_4
VT_{11}	0	0	0	0	0	P	1	P	Y_{17}-Y_{18}-Z_4
VT_{12}	0	0	0	0	0	1	P	P	Y_{20}-Y_{18}-Z_4

e) **Analiza ieşirii Z_5.**

$$Z_5 = X_8\bar{X}_9X_{10}X_{11}X_{12}\bar{X}_{13} + \bar{X}_8\bar{X}_9\bar{X}_{13}Y_3^* ;$$

Intrările / Vectorii de testare	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	Y_3^*	Ieşirea Z_5	Calea activată
VT_1	1	0	P	1	1	0	X0	P	X_{10}-Y_{16}-Y_{19}-Y_{20}-Y_{21}-Z_5
VT_2	1	0	1	P	1	0	X0	P	X_{11}-Y_{16}-Y_{19}-Y_{20}-Y_{21}-Z_5
VT_3	1	0	1	1	P	0	X0	P	X_{12}-Y_{19}-Y_{20}-Y_{21}-Z_5
VT_4	P	0	1	1	1	0	X0	P	X_8-Y_{20}-Y_{21}-Z_5
VT_5	1	0	1	1	$\bar{P}$	0	X1	$\bar{P}$	X_{12}-Y_{19}-Y_{20}-Y_{21}-Z_5
VT_6	0	P	0	0	0	0	X1	$\bar{P}$	X_9-Y_{14}-Y_{20}-Y_{21}-Z_5
VT_7	1	0	1	1	1	P	1	$\bar{P}$	X_{13}-Y_{22}-Y_{21}-Z_5
VT_8	0	0	0	0	0	0	P	P	Y_{20}-Y_{21}-Z_5

f) **Analiza ieşirii Z_6.**

$$Z_6 = \bar{X}_{13}X_{15}X_{17} + \bar{X}_{14}X_{15}X_{17} + \bar{X}_{15}X_{16}X_{17} ;$$

Intrările / Vectorii de testare	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	Ieşirea Z_6	Calea activată
VT_1	P	1	1	0	1	$\bar{P}$	X_{13}-Y_{23}-Y_{24}-Y_{25}-Y_{26}-Y_{27}-Z_6
VT_2	1	P	1	0	1	$\bar{P}$	X_{14}-Y_{23}-Y_{24}-Y_{25}-Y_{26}-Y_{27}-Z_6
VT_3	0	0	P	0	1	P	X_{15}-Y_{24}-Y_{25}-Y_{26}-Y_{27}-Z_6
VT_4	1	1	P	1	1	$\bar{P}$	X_{15}-Y_{28}-Y_{29}-Y_{25}-Y_{26}-Y_{27}-Z_6
VT_5	0	0	0	P	1	P	X_{16}-Y_{29}-Y_{25}-Y_{26}-Y_{27}-Z_6
VT_6	0	0	1	0	P	P	X_{17}-Y_{27}-Z_6

g) Analiza ieşirii Z_7.

$$Z_7 = X_{13}X_{16}X_{17}X_{18}X_{19}X_{20}X_{21} + X_{13}X_{16}X_{17}X_{18}X_{19}X_{20}X_{22} \quad ;$$

Intrările / Vectorii de testare	X_{13}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	Ieşirea Z_7	Calea activată
VT_1	1	1	1	1	1	1	P	0	P	X_{21}-Y_{36}-Y_{34}-Y_{32}-Y_{33}-Z_7
VT_2	P	1	1	1	1	1	1	0	P	X_{13}-Y_{33}-Z_7
VT_3	1	P	1	1	1	1	1	0	P	X_{16}-Y_{30}-Y_{31}-Y_{32}-Y_{33}-Z_7
VT_4	1	1	P	1	1	1	1	0	P	X_{17}-Y_{30}-Y_{31}-Y_{32}-Y_{33}-Z_7
VT_5	1	1	1	P	1	1	1	0	P	X_{18}-Y_{30}-Y_{31}-Y_{32}-Y_{33}-Z_7
VT_6	1	1	1	1	P	1	1	0	P	X_{19}-Y_{30}-Y_{31}-Y_{32}-Y_{33}-Z_7
VT_7	1	1	1	1	1	P	1	0	P	X_{20}-Y_{32}-Y_{33}-Z_7
VT_8	1	1	1	1	1	1	0	P	P	X_{22}-Y_{39}-Y_{34}-Y_{32}-Y_{33}-Z_7

h) Analiza ieşirii Z_8.

$$Z_8 = X_{20} + \bar{X}_{21}X_{22} + \bar{X}_{21}X_{23} \quad ;$$

Intrările / Vectorii de testare	X_{20}	X_{21}	X_{22}	X_{23}	Ieşirea Z_8	Calea activată
VT_1	P	0	0	0	P	X_{20}-Y_{35}-Y_{38}-Z_8
VT_2	0	0	P	0	P	X_{22}-Y_{39}-Y_{41}-Y_{37}-Y_{38}-Z_8
VT_3	0	P	1	0	$\bar{P}$	X_{21}-Y_{36}-Y_{37}-Y_{38}-Z_8
VT_4	0	0	0	P	P	X_{23}-Y_{40}-Y_{41}-Y_{37}-Y_{38}-Z_8

i) Analiza ieşirii Z_9.

$$Z_9 = \bar{X}_{16} + \bar{X}_{17} + \bar{X}_{18} + \bar{X}_{19} + \bar{X}_{20} + X_{22} + X_{23} + \bar{X}_{24}$$

Intrarile / Vectorii de testare	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{22}	X_{23}	X_{24}	Iesirea Z_9	Calea activata
VT_1	P	1	1	1	1	0	0	1	$\bar{P}$	X_{16}-Y_{30}-Y_{31}-Y_{43}-Y_{44}-Z_9
VT_2	1	P	1	1	1	0	0	1	$\bar{P}$	X_{17}-Y_{30}-Y_{31}-Y_{43}-Y_{44}-Z_9
VT_3	1	1	P	1	1	0	0	1	$\bar{P}$	X_{18}-Y_{30}-Y_{31}-Y_{43}-Y_{44}-Z_9
VT_4	1	1	1	P	1	0	0	1	$\bar{P}$	X_{19}-Y_{30}-Y_{31}-Y_{43}-Y_{44}-Z_9
VT_5	1	1	1	1	P	0	0	1	$\bar{P}$	X_{20}-Y_{43}-Y_{44}-Z_9
VT_6	1	1	1	1	1	P	0	1	$\bar{P}$	X_{22}-Y_{39}-Y_{41}-Y_{42}-Y_{43}-Y_{44}-Z_9
VT_7	1	1	1	1	1	0	P	1	P	X_{23}-Y_{40}-Y_{41}-Y_{42}-Y_{43}-Y_{44}-Z_9
VT_8	1	1	1	1	1	0	0	P	$\bar{P}$	X_{24}-Y_{44}-Z_9

Pentru localizarea defectelor se descrie schema cu ajutorul limbajului simbolic si aplicînd semnalele stabilite prin vectorii de testare, pe intrările schemei, se determină, cu ajutorul simulatorului, valoarea logică a fiecărui nod din schemă.

```
*D
INTRODUCETI PROGRAMUL

$DI1,Y2F1,Y3F3-
I2,Y1F1-
I3,Y1F2-
I4,Y1F4-
I5,Y1F5-
Y1F3,Y1F10-
Y1F6,Y1F9-
Y1F8,Y2F2,Y11F13-
Y2F3,D1-
I6,Y11F1,Y3F13,Y8F4-
I7,Y3F1,Y14F2-
Y3F2,Y11F2-
Y11F12,D2-
Y3F4,Y6F1-
I8,Y6F2,Y13F11,Y13F3-
Y6F3,Y14F3-
I9,Y3F5-
Y3F6,Y14F1,Y13F1,Y13F13-
Y14F5,Y8F3-
Y3F12,Y8F2-
I10,Y8F5,Y6F4-
Y8F6,Y3F11-
Y3F10,D3-
I11,Y6F5-
Y6F6,Y13F2,Y6F13-
Y13F5,Y6F10-
I12,Y6F12-
Y6F11,Y13F12-
Y13F9,Y6F9,Y7F1-
Y6F8,D4-
I13,Y3F9,Y9F1,Y7F13-
Y3F8,Y7F2-
Y7F3,D5-
I14,Y9F2-
Y9F3,Y8F13-
I15,Y8F1,Y4F1-
Y4F2,Y8F10-
I16,Y8F9,Y12F1-
Y8F8,Y4F3-
Y4F4,Y7F10-
I17,Y7F9,Y12F2-
Y7F8,D6-
I18,Y12F4-
I19,Y12F5-
Y12F6,Y4F5-
Y4F6,Y11F11,Y11F3-
I20,Y11F10,Y4F13,Y11F5-
I22,Y4F9-
Y4F8,Y9F4,Y9F10-
I21,Y4F11-
Y4F10,Y9F13,Y9F5-
Y9F6,Y11F9-
```

```
Y11P8,Y7P12-
Y7P11,O7-
Y4P12,Y10P1
I23,Y5P1-
Y5P2,Y9P9
Y9P8,Y9P12,Y5P3-
Y9P11,Y10P2-
Y10P3,O8-
Y5P4,Y11P4-
Y11P6,Y10P4
I24,Y10P5-
Y10P6,O9-
$EY1=CDB:486-
Y2,Y6,Y7=CDB:408-
Y3,Y4,Y5=CDB:404-
Y8=CDB:451-
Y9,Y10=CDB400-
Y11=CDB:411-
Y12=CDB:420-
Y13,Y14=CDB:474-
$F.
    .
    .
    .
INTRODUCETI VALORILE

*EI1,I2,I4,I5,I6,I8,I10,I11,I12,I14,I15,I16,I17,I18,I19,I20,I24=1.
*ET3,I7,I9,I13,I21,I22,I23=0.

AFISARE REZULTATE
-*
I001P01  1,
I002P01  1,
I003P01  0,
I004P01  1,
I005P01  1,
I006P01  1,
I007P01  0,
I008P01  1,
    .
    .
    .
    .

Y1   P03 1,
Y1   P06 0,
Y1   P08 1,
Y2   P03 1,
0001P01  1,
Y3   P02 1,
Y11  P12 1,
0002P01  1,
Y3   P04 0,
Y3   P06 1,
Y6   P03 0,
Y14  P05 0,
Y3   P12 0,
Y8   P06 0,
Y3   P10 1,
0003P01  1,
     .
     .
     .

NOTATII:
 INTRARI-I ; -IESIRI-O ; -CIRCUITE-Y ;
```

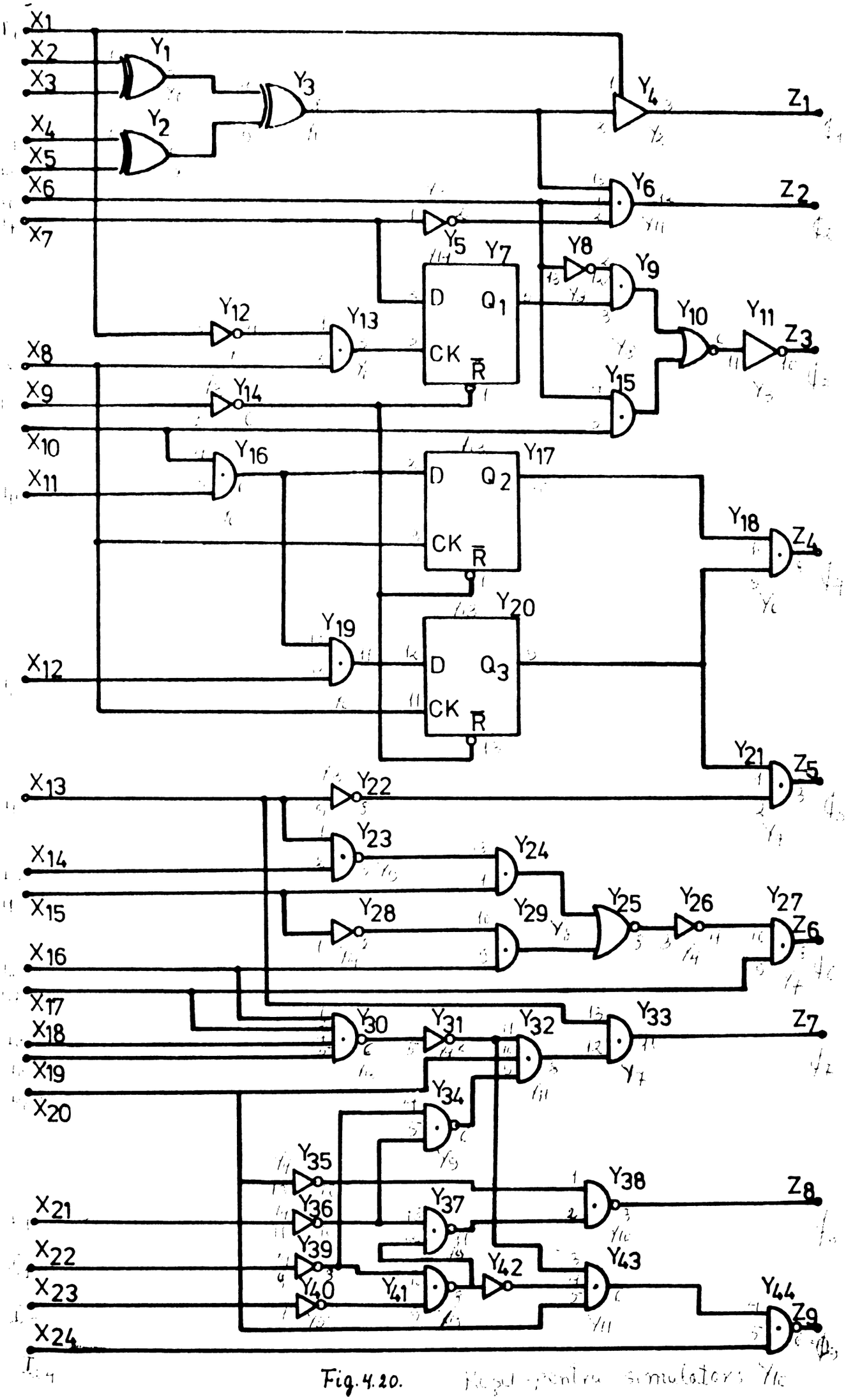

X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
X21
X22
X23
X24
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y8
Y9
Y10
Y11
Y12
Y13
Y14
Y15
Y16
Y17
Y18
Y19
Y20
Y21
Y22
Y23
Y24
Y25
Y26
Y27
Y28
Y29
Y30
Y31
Y32
Y33
Y34
Y35
Y36
Y37
Y38
Y39
Y40
Y41
Y42
Y43
Y44
D
Q1
Q2
Q3
CK
R
Z1
Z2
Z3
Z4
Z5
Z6
Z7
Z8
Z9

Fig. 4.20.

4.9. Concluzii

Concluziile ce se desprind din problemele tratate în acest capitol sînt următoarele :

a) Configuraţia sistemelor automate de testare este influenţată de tipul şi complexitatea elementelor care se verifică;

b) Sistemele automate de testare a plachetelor echipate sînt unele din cele mai complexe, iar realizarea lor implică o investiţie importantă de timp şi mijloace;

c) In realizarea sistemelor automate de testare se păstrează, în general, aceeaşi configuraţie modulară;

d) Tipul de conectare printr-o magistrală universală a tuturor modulelor sistemului asigură un înalt grad de flexibilitate, siguranţă în funcţionare, modularitate, toate acestea, la o viteză de transfer a informaţiilor ridicată;

e) Sistemul automat de testare prezentat asigură o testare funcţional-dinamică a plachetelor echipate, pe baza metodologiei stabilite în cap.2 şi 3, privind elaborarea vectorilor de testare;

f) Sistemele automate de testare sînt indispensabile în actuala etapă de dezvoltare, datorită complexităţii mereu crescînde a echipamentelor ce se realizează.

Cap.5. CONCLUZII FINALE

Integrarea unui număr mare de funcţii la nivelul unei componente, pe de o parte, şi necesitatea realizării de echipamente complexe, pe de altă parte, sînt elemente care impun evidenţierea etapei de punere în funcţiune şi evaluare a performanţelor produselor, ca una dintre cele mai importante ale procesului de producţie. O tehnologie de testare adecvată joacă un rol important în rezolvarea problemelor caracteristice acestei etape.

Prezenta lucrare tratează problema testării plachetelor echipate cu circuite logice SSI, MSI şi LSI.

Deoarece elementul constructiv de bază, din configuraţia unui produs, îl constituie placheta, este firesc ca şi problemele cele mai importante şi mai dificil de rezolvat, din punctul de vedere al testării, să apară la acest nivel, iar tratarea lor în lucrare să fie justificată şi oportună.

Analizînd diferitele tipuri de plachete echipate se stabilesc, în funcţie de complexitate şi număr de circuite, 4 tipuri de bază, evidenţiindu-se elementele caracteristice, din punctul de vedere al operaţiei de testare. Testarea este privită şi în ansamblul procesului de producţie, prin tehnologia de testare în flux. In acest sens se analizează diferitele tipuri de tehnologii de testare, cu particularităţile aferente, tipurile de defecte, verificările ce se efectuează şi unele elemente ce determină o strategie de testare optimă. Strategia de testare optimă este diferită de la produs la produs, în funcţie de caracteristicile fiecăruia, dar scopul urmărit este acelaşi şi anume, minimizarea costului şi timpului afectat operaţiei de testare, în condiţiile asigurării unei depanări adecvate. Eficacitatea operaţiei de testare este direct influenţată de metoda utilizată pentru construirea vectorilor de testare, modul de elaborare al programelor şi echipamentul de testare utilizat. In acest sens se analizează critic metodele existente, evidenţiindu-se avantajele şi dezavantajele acestora, în comparaţie cu metoda propusă de autor. Totodată se concluzionează faptul că un sistem automat de testare, datorită performanţelor ce le are, răspunde cel mai bine necesităţilor de testare ale plachetelor echipate cu circuite logice.

Acestea sînt problemele tratate în primul capitol al lucrării, urmărindu-se, astfel, justificarea abordării domeniului cu elementele sale caracteristice şi o prezentare introductivă a tuturor problemelor ce vor fi dezvoltate în capitolele următoare.

In capitolul doi se propune o metodă de testare a schemelor combinaţionale şi/sau secvenţiale sincrone, realizate cu circuite logice SSI, MSI şi LSI, utilizînd principiul activării căilor. In acest sens se elaborează un algoritm prin care se urmăreşte determinarea combinaţiilor de semnale, care aplicate pe intrările schemei, activează cel puţin o cale de la o intrare, la o ieşire a schemei. Combinaţia de semnale, care are această proprietate, poartă numele de vector de testare. Se poate afirma că intrarea respectivă controlează calea activată, astfel încît orice variaţie a valorii logice a semnalului aplicat pe acea intrare se observă la ieşire. Posibilitatea efectuării unui transfer pentru orice valoare logică, de la o intrare a schemei, prin circuitele care apar pe calea activată, pînă la ieşirea schemei şi modul de realizare a acestuia, în cazul fiecărui tip de circuit logic, se prezintă cu ajutorul MPA. Utilizînd MPA aferent unui anumit tip de circuit logic, se arată că pentru oricare din intrările sale (de date sau de comandă) sau plecînd din orice stare a sa, există o combinaţie de semnale, care aplicate pe celelalte intrări, activează cel puţin o cale pînă la o ieşire.

Realizarea MPA se prezintă pentru toate tipurile de circuite logice SSI, MSI şi LSI. Pentru cazul circuitelor secvenţiale MSI şi a circuitelor LSI se propune utilizarea aceluiaşi tip de descriere sub forma MPA, de tip registru sau numărător, cu posibilităţi de încărcare, particularizarea pentru un anumit circuit făcîndu-se după modul de specificare a capacităţii, a intrărilor de date, respectiv de comandă şi a ieşirilor de date, respectiv de control. Astfel se obţine o importantă reducere a memoriei solicitate pentru păstrarea diferitelor MPA, deoarece se reţine un singur MPA şi în plus, pentru fiecare circuit, se execută o rutină care-i specifică elementele caracteristice (intrări, ieşiri, capacitate etc.).

In cazul circuitelor LSI, datorită complexităţii şi reducerii substanţiale a punctelor de acces, activarea diferitelor module din configuraţie, pentru a se putea realiza un control de la o intrare accesibilă pînă la o ieşire accesibilă, este necesar să se facă gradat, ţinînd cont de posibilităţile de acces la

magistralele externe, pe care le au diferitele module. Pentru aceasta se elaborează un algoritm, care,plecînd de la descrierea instrucţiunilor circuitului LSI pe baza unui set de microoperaţii universale, realizează activarea gradată a fiecărui modul din configuraţia circuitului LSI. Odată realizată această activare se poate obţine şi o testare a fiecărui modul în parte. Din punctul de vedere al utilizatorului se cere numai descrierea instrucţiunilor circuitelor LSI, cu ajutorul setului de microoperaţii şi specificarea modulelor cu acces la magistralele externe. Algoritmul stabileşte toate modulele circuitului, le descrie sub forma MPA de tip registru sau numărător şi le ordonează astfel încît să poată fi verificate individual, modulele verificate şi declarate corecte utilizîndu-le pentru verificarea altora. Vectorii de testare, care se aplică pe intrările accesibile, se construiesc cu ajutorul algoritmului amintit anterior şi făcînd apel la descrierea circuitului LSI sub forma de MPA de tip registru sau numărător şi ordonate în vederea unei verificări gradate.

In acest fel s-a realizat o descriere unitară a fiecărui tip de circuit SSI, MSI şi LSI. Totodată se menţionează faptul că, existenţa unei descrieri de tipul MPA pentru un anumit tip de circuit, este foarte avantajoasă în cazul apelării la tehnica de testare "IN CIRCUIT", deoarece oferă o posibilitate de verificare rapidă şi sigură pentru orice circuit logic din configuraţia schemei realizată pe placheta care se verifică. Programul de testare, care se realizează în acest caz, este o copie fidelă a descrierii sub forma MPA (în special a TED) pentru circuitul care se verifică.

Urmărind preocupările şi rezultatele obţinute, pe plan mondial, prin utilizarea tehnicii de testare "IN CIRCUIT" se apreciază că, în scurt timp, aceasta va deveni una din cele mai utilizate şi mai avantajoase din punct de vedere al raportului performanţă/cost, astfel încît realizarea programelor de testare, pe baza descrierii circuitelor sub forma MPA, va deveni o necesitate absolută.

Algoritmul, cu ajutorul căruia se construiesc vectorii de testare, se execută pentru fiecare ieşire a schemei realizată pe plachetă. Pentru elaborarea vectorilor de testare se utilizează funcţia logică realizată la ieşirea respectivă. Această funcţie se scrie ţinînd cont de funcţiile logice realizate de fiecare circuit, a cărui comportare influenţează valoarea logică ce

se obţine la respectiva ieşire. Funcţiile logice aferente circuitelor se scriu pe baza TED al MPA, pentru circuitul în cauză. Odată obţinută funcţia realizată la ieşirea accesibilă, în raport cu intrările accesibile şi, eventual, stările interne ale circuitelor, este necesar să se rezolve acea ecuaţie obţinută prin egalarea funcţiei cu 1 sau cu 0, care are numărul minim de soluţii. Pentru aceasta se elaborează un algoritm, relativ simplu, cu ajutorul căruia se determină ecuaţia cu număr minim de soluţii înainte de rezolvare şi care, totodată, aduce ecuaţia la forma în care nu are soluţii multiple. In felul acesta se obţine o reducere considerabilă a numărului de calcule.

In funcţie de configuraţia schemei, vectorii de testare pot activa cîte o singură cale fiecare sau mai multe căi simultan. Pentru activarea, la un moment dat, a unei singure căi, se propune un algoritm prin care se introduc puncte de control suplimentare, pe căile activate simultan, astfel încît unele dintre acestea vor putea fi blocate la un moment dat. In afara operaţiei de testare punctele de control sînt ineficiente.

Modul de aplicare al metodei de testare se prezintă pe cîteva scheme combinaţionale şi secvenţiale.

Metoda permite elaborarea setului complet al vectorilor de testare, pe baza unui procedeu relativ simplu şi cu aplicabilitate directă în practică. Aceste elemente caracteristice sînt valabile pentru scheme combinaţionale şi secvenţiale sincrone iredundante. Totodată, metoda oferă informaţii minore asupra legăturilor de reacţie.

In capitolul trei se abordează problema elaborării automate a vectorilor de testare. Pentru aceasta se analizează, mai întîi, configuraţia sistemului de programe de testare, care intră în dotarea unui sistem automat de testare, insistîndu-se, în special, asupra limbajelor orientate spre testare. In acest sens se concluzionează faptul, că soluţia cea mai bună este utilizarea unui limbaj, care să permită elaborarea programelor de testare, bazîndu-se pe un control orientat spre elementul sub test, dar să asigure conservarea imediată şi completă a controlului resurselor sistemului. Totodată se apreciază faptul că, realizarea compilării în două faze, a programelor scrise în limbaje orientate spre testare, oferă o soluţie de compromis, dar care încearcă să ţină cont de toate problemele legate de alocarea resurselor.

Datorită particularităţilor şi complexităţii operaţiei de

testare, este mult mai indicată utilizarea limbajelor orientate spre testare, în locul celor cu orientare generală.

Pentru descrierea schemelor realizate pe plachete se elaborează un limbaj simplu şi orientat, uşor de asimilat, apropiat de limbajul utilizatorilor de circuite logice. Cu ajutorul lui se poate descrie, în memoria echipamentului de calcul, orice schemă logică combinaţională şi/sau secvenţială, cu sau fără reacţii, care are în componenţa sa şi tranzistoare, rezistenţe sau condensatoare.

Analizorul sintactic al limbajului verifică fiecare legătură descrisă, elimină informaţiile suplimentare (separatori, caractere speciale etc.), reţinînd, sub forma unor tabele, numai informaţiile utile, ce descriu corect legăturile. Se semnalează tipul şi locul fiecărei erori sintactice, precum şi numărul total al erorilor.

Fiecare circuit logic combinaţional sau secvenţial este descris în biblioteca de circuite, unde se reţin, pentru fiecare caz în parte, parametrii circuitului logic, tipul pinilor şi funcţia logică realizată. Făcînd apel la descrierea din bibliotecă se verifică utilizarea corectă a fiecărui circuit din schemă.

Pentru simularea schemei descrise s-a elaborat un procesor de simulare, realizat pe bază de tabele şi care permite simularea a patru stări logice. Acesta preia informaţiile verificate de analizorul sintactic, le aranjează într-o ordine prestabilită, în tabele, astfel încît explorarea secvenţială a acestora să permită determinarea valorii logice a oricărei ieşiri, în funcţie de semnalele aplicate pe intrări. In acest fel s-a creat o imagine virtuală, corectă, a schemei reale, realizată pe plachetă, imagine care poate fi atacată, la intrare, cu aceeaşi secvenţă de stimuli ca şi schema de pe plachetă. Aceasta constituie un avantaj important în faza de localizare a defectelor pe plachetă, deoarece imaginea simulată oferă, în orice moment, starea corectă a tuturor nodurilor schemei, iar prin comparaţie cu starea reală de pe plachetă, se poate localiza rapid un anumit defect. Deoarece, în biblioteca de circuite se află memorat şi timpul de propagare aferent fiecărui circuit, procesorul de simulare permite şi depistarea hazardului, a impulsurilor parazite sau a condiţionărilor greşite.

Totodată, prin aplicarea vectorilor de testare asupra in-

trărilor schemei simulate se obţine valoarea corectă a fiecărei ieşiri şi astfel se poate realiza programul complet de testare a plachetei echipate.

Procesorul de defecte este utilizat pentru injectarea de defecte de tipul f1 sau f0 în schema simulată şi astfel, în urma aplicării vectorilor de testare, pe intrări, să se stabilească eficacitatea acestora. O problemă foarte importantă pentru reducerea numărului total de defecte este stabilirea defectelor echivalente. S-au stabilit două tipuri de clase de echivalenţă, iar în faza de execuţie se abordează un singur defect din fiecare clasă. Astfel se reduce, considerabil, numărul de calcule.

In concluzie se poate afirma, că utilizarea unui procesor de simulare permite realizarea, în mod automat, dar cu un înalt grad de eficienţă, a programelor de testare. Totodată procesorul de simulare oferă avantaje importante în faza de localizare a defectelor. Se menţionează faptul că procesorul de simulare poate fi utilizat şi pentru evaluarea performanţelor, în faza de proiectare a schemelor.

Sistemul de programe de testare elaborat este destinat schemelor realizate cu circuite logice, iar eficienţa lui este direct influenţată de tipul şi complexitatea schemei.

In capitolul patru se prezintă organizarea unui sistem automat de testare, conceput de autor. Urmărind configuraţia diferitelor sisteme de testare se evidenţiază tipurile de module ce pot să apară, caracteristicile şi destinaţia fiecăruia. Interconectarea diferitelor module ale unui echipament de testare se realizează printr-o magistrală universală. Datorită avantajelor ce le oferă s-a adoptat magistrala universală, caracteristică standardului IEEE 488-1975. Se propune utilizarea aceleiaşi magistrale, împreună cu interfaţa aferentă, pentru toate modulele echipamentului, indiferent de tip (numeric, analogic, matricea cu relee etc.), spre deosebire de echipamentele existente în momentul de faţă, care utilizează o magistrală universală standardizată, eventual pentru cuplarea unor aparate. In acest caz schimbul de informaţii se face în mod identic cu toate modulele, deci sistemul de programe de testare nu va conţine opţiuni distincte pentru modulele conectate în moduri diferite, ceea ce contribuie mult la simplificarea acestuia. Totodată, realizarea din punct de vedere electronic a tuturor interfeţelor, utilizate la interconectare, este perfect identică, ceea ce oferă avan-

taje importante în fazele de realizare tehnologică şi de punere în funcţiune. Datorită posibilităţilor de interschimbabilitate, existente în acest caz, pentru toate modulele apar avantaje la punerea în funcţiune a întregului sistem automat de testare, iar fiabilitatea şi siguranţa în funcţionare, la nivelul sistemului, se îmbunătăţesc considerabil. Se menţionează faptul că viteza de prelucrare şi transfer a informaţiilor pe magistrala universală rămîne ridicată, fiind limitată superior de posibilităţile echipamentului de calcul şi de modulele care participă la schimb.

In această lucrare se prezintă configuraţia unui sistem automat destinat testării funcţional-dinamice a plachetelor echipate cu circuite logice SSI, MSI şi LSI, aparţinînd familiilor TTL şi CMOS. Se insistă asupra modulelor de testare, de comandă şi a modului de interconectare prin magistrala universală, elemente realizate de autor.

Operaţia de testare se desfăşoară în două etape. In prima etapă, cu ajutorul modulului de testare se pregătesc semnalele ce urmează a fi aplicate pe pinii plachetei ce se verifică, în configuraţia cerută de aceasta, se stabileşte tipul pinilor, modul de testare, modul de afişare etc. In etapa a doua se comandă aplicarea sincronă şi simultană a semnalelor pe pinii de intrare ai plachetei şi înregistrarea răspunsurilor obţinute pe pinii de ieşire, după un timp prestabilit şi, de asemenea, sincron cu un semnal de tact sau cu un semnal de comandă. Utilizarea modului de lucru în două etape a fost necesară, deoarece informaţia care se extrage din echipamentul de calcul are un format de lungime fixă, lungime care, în majoritatea cazurilor, este mult mai mică decît cea cerută de placheta care se testează. In general, fiecărui pin îi corespunde un bit de informaţie. In acest caz, dacă informaţia extrasă din echipamentul de calcul s-ar aplica direct pe pinii plachetei, schema logică ar trece prin stări intermediare dificil de controlat, în faza de testare.

Realizînd o aplicare simultană şi sincronă a semnalelor pe toţi pinii de intrare ai plachetei şi înregistrarea controlată a răspunsurilor, se asigură şi o verificare a comportării dinamice a schemei realizată pe plachetă, prin analiza timpului de propagare a semnalelor prin schemă.

Pentru obţinerea unei viteze de testare ridicată, necesară mai ales în cazul testării plachetelor echipate şi cu circuite LSI, s-a introdus cîte un bloc de memorie pentru înmagazina-

rea unui număr dat de vectori de testare şi unul pentru înregistrarea răspunsurilor, în acest caz. Astfel se obţine o viteză de aplicare a semnalelor pe pinii plachetei, limitată superior numai de timpul de acces al blocurilor de memorie.

Intregul echipament de testare este realizat modular, cu posibilitate de interschimbabilitate a tuturor plachetelor de acelaşi tip şi urmărindu-se, în principal, obţinerea unei fiabilităţi şi a unei siguranţe sporite în funcţionare.

Posibilităţile de autotestare, atît din punct de vedere hard, cît şi soft, ce le deţine, asigură o depanare rapidă şi sigură.

Modul de realizare şi performanţele obţinute permit încadrarea sistemului automat de testare prezentat în generaţia3,5, aferentă acestui domeniu.

In final, autorul îşi exprimă speranţa că a realizat o metodologie completă privind testarea plachetelor echipate cu circuite logice şi anume : elaborarea unei metode de testare,realizarea unui sistem destinat elaborării automate a programelor de testare, bazate pe metoda concepută şi realizarea unui sistem automat care permite testarea plachetelor echipate, cu ajutorul programului construit.

Principalele contribuţii aduse de autor în preţenta lucrare sînt următoarele :

a) - elaborarea unei metode de testare a schemelor logice combinaţionale şi secvenţiale sincrone, care cuprinde :
   - algoritmul de activare a căilor de la fiecare intrare a schemei, la o ieşire;
   - descrierea fiecărui tip de circuit logic sub forma de MPA, astfel încît orice variaţie a valorii logice a unui semnal aplicat pe o intrare, să poată fi pusă în evidenţă la cel puţin o ieşire.

   Ca elemente ajutătoare s-au elaborat următorii algoritmi:
   - algoritm pentru stabilirea şi activarea modulelor din configuraţia unui circuit LSI;
   - algoritm pentru eliminarea soluţiilor multiple ale unei ecuaţii logice;
   - algoritm pentru introducerea punctelor de control suplimentare;

b) - construirea unui sistem de programe de testare, care cuprinde :

- elaborarea unui limbaj de descriere a schemelor realizate cu circuite logice;
- construirea unui simulator destinat schemelor realizate cu circuite logice;
- construirea unor programe, care permit generarea automată a vectorilor de testare, întocmirea programelor de testare şi verificarea eficacităţii acestora;

c) - elaborarea unui sistem automat de testare a plachetelor echipate cu circuite logice SSI, MSI şi LSI, pentru care s-a realizat :
- stabilirea unei configuraţii generale, la nivel de modul, pentru orice sistem de testare;
- conceperea modulelor de testare şi de comandă din configuraţia sistemului automat de testare;
- introducerea şi realizarea conceptului de magistrală universală pentru toate modulele echipamentului de testare.

Pe întreaga perioadă de pregătire a prezentei lucrări de doctorat, autorul a lucrat, pe bază de contracte, împreună cu Institutul de cercetări pentru automatizări, tehnică de calcul şi telecomunicaţii (ICPATCT), filiala Cluj-Napoca. Rezultatele obţinute şi prezentate în capitolele lucrării s-au materializat în realizarea unor sisteme automate de testare, utilizate în momentul de faţă în producţie, la fabricile : Intreprinderea de electronică industrială şi automatizări Cluj-Napoca, Fabrica de elemente de automatizări Bucureşti, Electrotehnica Bucureşti şi Electromagnetica Bucureşti. De asemenea unele din sistemele realizate au fost apreciate elogios la expoziţiile internaţionale de la Brighton (Anglia), Wiesbaden (R.F.Germania), Paris (Franţa) şi Moscova (U.R.S.S.), fiind oferite în momentul de faţă pentru export. In prezent se realizează un astfel de sistem automat de testare pentru R.D.Germană.

Autorul îşi exprimă întreaga gratitudine conducerii institutului central, precum şi a filialei din Cluj-Napoca, pentru încrederea şi sprijinul acordat.

Sper ca modestele rezultate obţinute să servească pentru cercetări şi realizări ulterioare, cu performanţe în continuă evoluţie.

BIBLIOGRAFIE

A1. Abramovici,M.,Breuer,M.,A. - Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis. IEEE Transactions on Computers, vol.C-29, iunie 1980.

A2. Akers,S.,B. - A Logic System for Fault Test Generation. IEEE Transactions on Computers, vol.C-25, iunie 1976.

A3. Akers,S.,B. - Binary Decision Diagrams. IEEE Transactions on Computers, vol.C-27, iunie 1978.

A4. Angheloiu,I., Mincu,I., Györfi,E., Ghiţă,Al. - Introducere în sisteme tehnice mari. Editura Militară, Bucureşti, România, 1980.

A5. Antonescu,Al. - Tehnologii de testare în întreprinderile C.I.E.T.A. Studiu IPA. 1980.

B1. Batni,R., Kime,C. - A Module Level Testing Approach for Combinational Networks. IEEE Transactions on Computers, vol.C-25, iunie 1976.

B2. Beverly,N. - Interactive Simulator of IEEE-488. Test, vol. III, decembrie 1981.

B3. Bluestone,A. - Logical Environment Comparison Testing Handles Complex LSI Devices. Computer Design, aprilie 1979.

B4. Breuer,M.,A., Friedman,A.,D. - Functional Level Primitives in Test Generation. IEEE Transactions on Computers, vol. C-29, martie 1980.

B5. Budkowski,S. - Some Problems in verifying Microprograms. Proceedings of the Symposium on Microcomputer and Microprocessor Application, Budapesta, Ungaria, 1979.

C1. Cerny,E. - Controllability and Fault Observability in Modular Combinational Circuits. IEEE Transactions on Computer, vol.C-27, octombrie 1978.

C2. Chicoix,C., Pedoussant,J., Giambiasi,N. - An Accurate Time Delay. Model for Large Digital Network Simulation. IEEE Transactions on Computers.

C3. Chang,H.,Y., Manning,E., Metze,G. - Fault Diagnosis of Digital Systems. John Wiley, 1970.

C4. Covaciu,F. - Tehnologie de control globală a produselor din Intreprinderea Electrotehnica Bucureşti. Studiu IPA, 1979.

D1. Dancea,I. Calculatoare numerice. Editura didactică şi pedagogică, Bucureşti, Romania, 1975.

D2. Dancea,I. - Microprocesoare. Arhitectură internă. Programare. Aplicaţii. Editura Dacia, 1979.

F1. Forest,C. - Improve Testability of Logic Design. Membrain, Anglia, 1974.

F2. Friedman,A.,D., Menon,P.,R. - Fault Detection in Digital Circuits. Prentice Hall, 1971.

G1. Goundan,A., Hayes,J.,P. - Partitioning Logic Circuits so Maximize Fault Resolution. IEEE Transactions on Computers, vol.C-23, aprilie, 1974.

H1. Hammer (Ivănescu),P.,L., Rudeanu,S. - Boolean Methods in Operations Research and Related Areas. Springer Verlag, Berlin, Germania, 1968.

H2. Hartenstein,W. - LSI Chip Design : from evolution to revolution. Proceedings of the Symposium on Microcomputer and Microprocessor Application. Budapesta, Ungaria, 1979.

H3. Hayes,J. - Generation of Optimal Transition Count Tests. IEEE Transactions on Computers, vol.C-27, ianuarie, 1978.

H4. Hayes,J. - Transition Count Testing of Combinational Logic Circuits. IEEE Transactions on Computers, vol.C-25, iunie, 1976.

H5. Hăngănuţ,M., Negru,O., Leţia,I., Antonescu,Al. - Sisteme de testare. Buletinul ştiinţific IPCN, Cluj-Napoca, Romania, 1977.

H6. Hăngănuţ,M., Negru,O., Damian,D. - Sistem de testare cu calculator. Simpozionul naţional de creativitate tehnologică, Cluj-Napoca, Romania, 1978.

H7. Hăngănuţ,M., Negru,O. - Testor de plachete, Simpozionul de informatică, Cluj-Napoca, Romania, 1976.

H8. Hopf,T.,A. - Universal Board System. Test, vol.III, septembrie, 1981.

H9. Huang,H.,H. - MSI and LSI Impact on Digital Systems Testing. IEEE Transactions on Computer.

J1. Jenssen,K. - Production Line Testing of Microprocessor Based Products. Test, vol.III, decembrie, 1981.

K1. Keiner,W.,L. - Functional Testing. A User Looks at Logic Simulation. IEEE Transactions on Computers.

K2. Knowles,R. - Automatic Testing Systems and Applications. McGraw-Hill Book Company, 1976.

K3. Kohavi,Z. - Finite State Machines. John Wiley, 1971.

K4. Kohavi,Z., Berger,I. - Fault Diagnosis in Combinational Tree Networks. IEEE Transactions on Computers, vol.C-24, decembrie, 1975.

K5. Kosek,B. - An Integrated Hardware-Software. Debugging System Based on the Forced Instruction Method. Proceedings of the Symposium on Microcomputer and Microprocessor Application, Budapesta, Ungaria, 1979.

K6. Kovacs,M. - Tehnical and Economical Problems of LSI. Proceedings of the Symposium on Microcomputer and Microprocessor Application, Budapesta, Ungaria, 1979.

K7. Ku,C.T., Masson,G.,M. - The Boolean Difference and Multiple Fault Analysis. IEEE Transactions on Computers, vol. C-24, ianuarie, 1975.

L1. Lăzăroiu,D.,F., Bălan,M., Constantinescu,F.,T. - Microprocesor Testing Methods. Symposium on Controls Systems and Computer Science, Bucureşti, Romania, 1979.

L2. Lee,C.,S. - Modern Switching Theory and Digital Design. Prentice Hall, 1978.

L3. Leţia,I.,A., Pusztai,K., Negru,O. - Procesor orientat pentru testare. Sesiunea ştiinţifică IPCN, Cluj-Napoca, Romania, 1978.

I4. Loughry,D., Allen,S. - IEEE Standard 488 and Microprocessors Synergism. Proceedings of IEEE vol.66 nr.2, februarie, 1978.

M1. Matthews,N.,O. - Introduction to Automated Testing. Network Publication, Anglia, 1974.

M2. Morris,R.,L., Miller,J.,R. - Proiectarea cu circuite integrate TTL. Editura tehnică, Bucureşti, Romania, 1974.

N1. Negru,O., Gavrea,G. - Testarea Circuitelor LSI. Simpozionul I.E.I.A. Cluj-Napoca, Romania, 1981.

N2. Negru,O., Serban,D., Găvrea,V. - Simularea circuitelor numerice. Symposium on Controls Systems and Computer Science. Bucureşti, Romania, 1981.

N3. Negru,O. - Problemele testării în industria modernă de echipament de calcul. Referat de doctorat. Institutul Politehnic "Traian Vuia" Timişoara, 1978.

N4. Negru,O. - Organizarea unui sistem de testare cu calculator.

Referat de doctorat. Institutul Politehnic "Traian Vuia", Timişoara, 1979.

O1. Osborne,A. - An Introduction to Microcomputers. Osborne/Mc Graw-Hill, U.S.A. 1980.

O2. Osborne,A. - 8080 Programing for Logic Design. A. Osborne Incorporated, U.S.A., 1976.

P1. Parasch,G.,J. - Development and Application of a Designer Oriented Cyclic Simulator. IEEE Transactions on Computers.

P2. Petrescu,A., Zervos,C., Tepuş,N. - Limbaj de simulare a circuitelor logice. Institutul Politehnic, Bucureşti, Romania.

P3. Petrescu,M., Giumale,C.,A., Dumitru,P., Popescu,T. - Experimente privind realizarea unor terminale cu afişare pe tub catodic. Probleme de automatizare, vol.II, Editura Academiei, Romania, 1979.

P4. Popiel,J., Rosinski,A.,T. - Multilevel Simulation of LSI Digital Circuits. Proceedings of the Symposium on Microprocessor Application, Budapesta, Ungaria, 1979.

R1. Rogojan,Al. - Testor de baterii. Simpozionul de informatică. Cluj-Napoca, Romania, 1978.

R2. Rogojan,Al. - Calculatoare numerice. Institutul Politehnic "Traian Vuia" Timişoara, Romania, 1973.

R3. Rozeboom,R.,W. - An Implementation of Computer Aided Test Generation Techniques. IEEE Transactions on Computers.

R4. Rozeboom,R.,W. - Current Problems Related to LSI Functional Testing. IEEE Transactions on Computers.

S1. Sabniw,L. - Multi-Defect Real Time Diagnosis using a Single Pin Probe. IEEE Transactions on Computers.

S2. Stoffers,K.,E. - Test Set for Combinational Logic. The Edge Tracing Approach. IEEE Transactions on Computers, vol.C-29, august, 1980.

S3. Szygenda,S.,A., Lekkos,A.,A. - Integrated Techniques for Functional and Gate-Level Digital Logic Simulation. IEEE Transactions on Computers.

T1. Takahashi,Y., Rabins,M., Auslander,D.,M. - Control and Dynamic Systems. Addison - Wesley Publishing Company, U.S.A., 1970.

T2. Thatte,S.,M., Abraham,J.,A. - Test Generation for Microprocessors. IEEE Transactions on Computers, vol.C-29, iunie 1980.

T3. Thompson,E.,W., Szygenda,S.,A., Billawalo,N., Pierce,R. - Timing Analysis for Digital Fault. Simulation Using Assignable Delays. IEEE Transactions on Computers.

U1. Ulrich,E.,G. - Non-Integral Event Timing for Digital Logic Simulation. IEEE Transactions on Computers.

U2. Ulrich,E.,G., Baker,T. - The Concurrent Simulation of Nearly Identical Digital Networks. IEEE Transactions on Computers.

V1. Vaughn,G. - Cdalgo - A Test Pattern Generation Program. IEEE Transactions on Computers.

V2. Verma,J.,P., Selove,D.,M., Tessier,J.,N. - Automatic Test-Generation and Test Verification of Digital Systems. IEEE Transactions on Computers.

V3. Vlăduţiu,M., Cotarcă,M., Bulzu,F. - Programs Elaboration for Functional Testing of Microprocessors. Symposium on Control Systems and Computer Science, Bucureşti, Romania, 1979.

W1. Wang,D.,T. - An Algoritm for the Generation of Test Set for Combinational Logic Networks. IEEE Transactions on Computers, vol.C-24, iunie, 1975.

W2. Wilcox,P., Rombeek,H. - F/Logic - An Interactive Fault and Logic Simulator for Digital Circuits. IEEE Transactions on Computers.

Y1. Yau,S.,S., Yang,S.,C. - Multiple Fault Detection for Combinational Logic Circuits. IEEE Transactions on Computers, vol.C-24, martie, 1975.

Z1. Zervos,C., Athanasiu,I. - Fault Detection in Sequential Logic Circuits Using the State Register. Buletinul I.P. Bucureşti, 1981.

Z2. Zervos,C., Athanasiu,I. - Detectability of Logical Faults in Sequential Logic Circuits. Buletinul I.P. Bucureşti, 1981.

Z3. Zaks,R., Lesea,A. - Techniques d'interface aux microprocesseurs. Sylex, 1980.

*1. A designer's Guide to Signature Analysis. Application Note 222, Hewlett Packard, U.S.A. 1980.

*2. Adar DR 12/25. Adar Inc., USA, 1979.

*3. Application Articles on Signature Analysis. Application Note 222-2, Hewlett Packard, USA, 1980.

*4. Automatic Testing 80. Conference Proceedings. Paris, Franţa, 1980.

*5. Automatic Testing 81. Conference Proceedings. Brighton, Anglia, 1981.
*6. Capable Test Systems. Application Note 1o7. Computer Automation, USA, 1978.
*7. Capable 4000 Series. Test Logic. Simulation Systems. Computer Automation, USA, 1979.
*8. CAPS/APG Equipped Gen Rad Test Systems. Gen Rad, USA, 1978.
*9. Circuit Designers Can Make Digital Boards Easier to Test. Teradyne, USA, 1978.
*1o. Data Catalog. Intel, USA, 198o.
*11. Digital IC Test System. Hewlett Packard, USA, 1979.
*12. DTS-7o Digital Test System. Hewlett Packard, USA, 1978.
*13. DTS-7o ECL Test System. Hewlett Packard, USA, 198o.
*14. DTS-7o Digital Test System. Designing Digital Circuits for Testability. Application Note 21o-4, Hewlett Packard, USA, 1979.
*15. Electronics Test. Octombrie, 1979.
*16. FTC/3. International Symposium on Fault-Tolerant Computing. Palo Alto, USA, 1972.
*17. GR 1792 A thru D and GR 1797 Test Systems. Gen Rad, USA, 1977.
*18. GR 1795 Logic-Circuit Test System. Gen Rad, USA, 1977.
*19. Gen Rad Application Notes 1-7. USA, 1977.
*2o. Graduate Data Pack. Marconi Instruments, Anglia, 1979.
*21. Hewlett Packard Journal, mai 1977.
*22. Hewlett Packard Journal, martie 1979.
*23. L135 LSI Board Test System. Teradyne, USA, 1978.
*24. Lasar. Electronic Circuit Design Verification and Test System. UCC Anglia, 1979.
*25. Learning Probe. Membrain, Anglia, 1975.
*26. M 6800 Microprocessor. Applications Manual. Motorola, USA, 1975.
*27. MB 2420. Automatic Test Systems. Membrain, Anglia, 1977.
*28. Membrain Automatic Test Equipment. Membrain Schlumberger, Anglia, 1981.
*29. MICA 5000. Manufacturing In-Circuit Analyzer. Computer Automation, USA, 1979.
*3o. Microcomputer Components. Motorola, USA, 1979.
*31. 9000 Series Micro-System Troubleshooters. Fluke, USA, 1981.
*32. MX-17. LSI Test System. Adar, USA, 198o.

*33. New Electronics. Special Supplement, noiembrie 1979.
*34. Proceedings of the ACM-IEEE Design Automation Workshop. Dallas, USA, 1972.
*35. Test Programming GR Digital/Analog Test Systems. Gen Rad, USA, 1977.
*36. Testaid III. Hewlett Packard, USA, 1976.
*37. Teradyne Automatic Test Systems, USA, 1977.
*38. 3040 A Logictester for uP Boards. Bulletin Nr.1o5 Fluke, USA, 1977.
*39. The Independent UK Testing Laboratory. MTL, Anglia, 1981.
*4o. Type 1792 A and 1792 B Logic Test Systems General Radio, USA, 1973.
*41. Gen Rad Automatic Test Systems, USA, 1978.
*42. Fluke Automatic Test Systems, USA, 1979.
*43. Lorlin Automatic Test Systems, USA, 1979.
*44. Siemens Automatic Test Systems, R.F.G., 198o.
*45. In Circuit Emulator 8o. Intel, SUA, 1979.
*46. IEEE Standard 416. Atlas. IEEE, 1978.
*47. Manual de utilizare M18. Fabrica de calculatoare Bucureşti, 198o.
*48. Manual de utilizare Coral. Fabrica de calculatoare Bucureşti, 1981.
*49. Standard IEEE 488. IEEE, 1978.
*5o. Rational automatic measurements using IEC-BUS test assemblies. Rohde-Schwarz, 198o.
*51. The missing links. IEEE 488 Bus. ICS 1981.
*52. Accessing the 3582A memory with HP-IB. Hewlett Packard application note 245-4, 1979.
*53. Improving measurements in engineering and manufacturing. Hewlett Packard, SUA, 1976.