MINISTERUL EDUCATIEI SI INVATAMINTULUI
INSTITUTUL POLITEHNIC "TRAIAN VUIA" TIMISOARA
FACULTATEA DE ELECTROTEHNICA

ing. MARIAN GH.DANCAU

PROBLEME ALE TESTARII INDUSTRIALE A UNITATILOR DE MEMORIE RAM CU CIRCUITE INTEGRATE

- Teza de doctorat -

Conducator stiintific:
prof.dr.ing. EUGEN POP

- 1983 -

P R E F A T A

Lucrarea de faţă se înscrie în preocupările generale de ridicare a productivităţii muncii şi de creştere a fiabilităţi şi coeficientului de disponibilitate a sistemelor de calcul. Ea este o continuare a activităţii depuse de autor în perioada de peste patru ani cît a fost încadrat la Fabrica de memorii electronice şi componente pentru tehnica de calcul din Timişoara. In cuprinsul acestei lucrări autorul abordează atît domeniul elaborării secvenţelor de stimulare a memoriilor în vederea creşterii eficienţei testării cît şi automatizarea experimentului de testare propriu-zis.

Pe toată durata elaborării tezei am beneficiat de îndrumările competente, pline de răbdare şi înţelegere ale prof.dr. ing.Eugen Pop, căruia îi aduc pe această cale cele mai respectuase mulţumiri.

Tin să mulţumesc în mod deosebit conducerii filialei ITC Timişoara şi a Fabricii de memorii electronice pentru sprijinul acordat pe toată durata elaborării acestei lucrări.

Pentru discuţiile fructuoase pe care le-am purtat datorez calde mulţumiri întregului colectiv al filialei IPA din Cluj Napoca.

Datorez mulţumiri colectivului care a elaborat echipamentele de testare MOSTEST-03, R.Telescu, N.Balmez, M.Cotarcă, E.Gherman, M.Rimbasiu şi D.Vlăduţiu, de la filiala ITC Timişoara, precum şi colegilor mei A,Boboc, S.Dordea, A.M.Huschitt, M.Ivănescu, M.Popescu, M.Radiţă, A.Toma şi celorlalţi ingineri de la Fabrica de memorii electronice. Pentru sprijinul şi contribuţia deosebită adusă la elaborarea programului ADRDIA aduc mulţumiri colegului meu mat.I.Despi de la Centrul de calcul al ITC Timişoara.

De asemenea, datorez cele mai calde mulţumiri, în primul rînd colegilor mei dr.ing.I.Naforniţă şi dr.ing.Vlăduţiu pentru îndelungatele discuţii purtate, ajutorul şi sprijinul moral de care am beneficiat din partea lor pe durata elaborării tezei.

Autorul

C U P R Í N S

CHECKERBOARD (A.2-4), MARCH (A.2-4), MOVI (A.2-5), GALCOL (A.2-5), GALROW (A.2-7), GALDIA (A.2-7), WALKPAT (A.2-1o), GALPAT (A.2-1o), MASEST (A.2-11), DIAPAT (A.2-12), WAKOL (A.2-14), DUAL WAKCOL (A.2-15), HAFCOL (A.2-17), HAMPAT (A.2-17), DIST WALKING (A.2-19), X-Y COMPL.(A.2-2o), GWR (A.2-22), PARITY (A.2-22), CHEKCOL (A.2-23), Knaizuk-Nair (A.2-26), MILPAT (A.2-28), NAIR (A.2-29), SUK (A.2-31).

3. Anexa A3 Diagramele Shmoo.
4. Anexa A4 Echipament de testare automată a plachetelor de memorie MOSTEST-03D.
5. Anexa A5.
6. Anexa A6.

CAPITOLUL 1.

INTRODUCERE

Am asistat în ultimii ani la o evoluţie spectaculoasă şi fără precedent a circuitelor integrate, această evoluţie încadrîndu-se în ceea ce prof.Mihai Drăgănescu numea "a doua revoluţie industrială".

In aceste condiţii de pătrundere a calculatoarelor în mai toate domeniile, începînd din anul 197o, cînd firma IBM a lansat primul calculator cu memorii semiconductoare ponderea feritelor a continuat să scadă ajungînd astăzi la sub 4% (fig. 1.1).

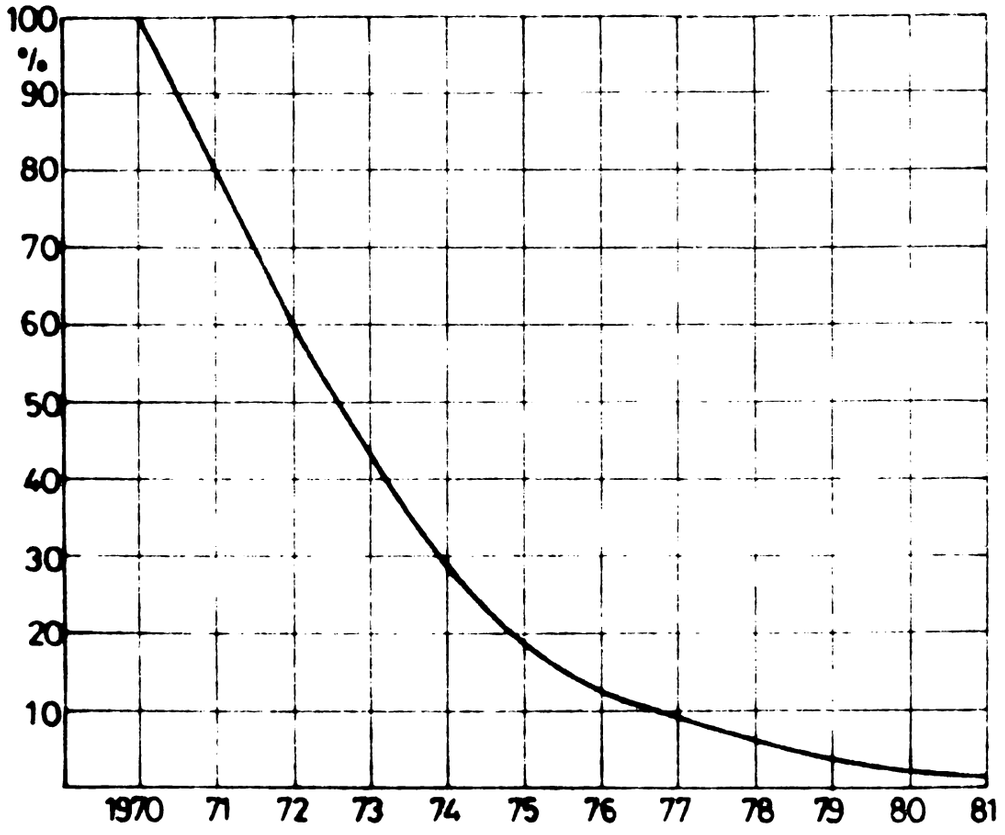


Fig. 1.1 Ponderea memoriilor cu ferite [BATR 81]

Fig.1.1. Ponderea memoriilor cu ferite [BATR81]

Locul acestora a fost luat în ritm galopant de memoriile cu semiconductoare, în special MOS, în paralel cu perfecţionarea lor. Cele mai optimiste previziuni din anul 1975 - 16 Kb în 1985 [MART75] au fost depăşite.

Se prevede că în perioada 1981-1985 piaţa mondială necaptivă de C.I. de memorie va avea o creştere de 26% fiind depăşită doar de microprocesoare cu 3o% [ELIN81] (fig.1.2).

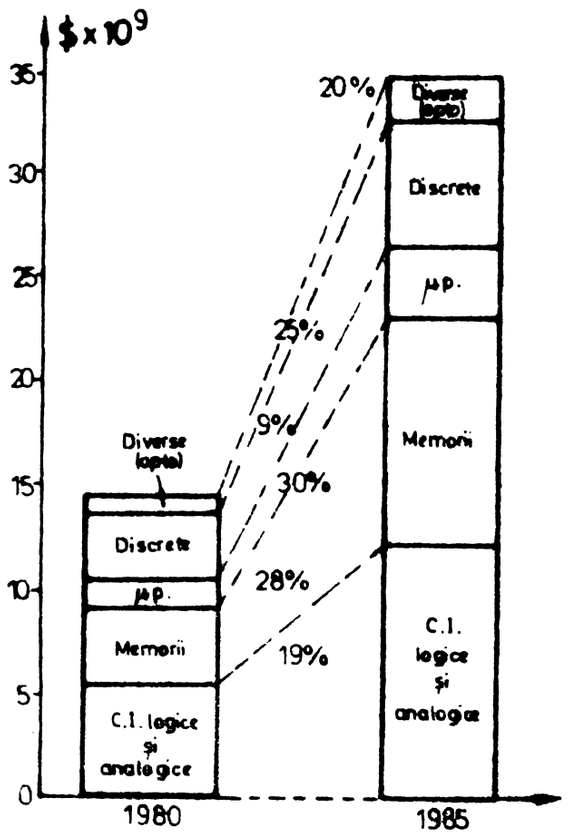


Fig.1.2. Prognoză pentru piaţa mondială de semiconductoare (necaptivă)

Fig.1.2. Prognoză pentru piaţa mondială de semiconductoare.

Ponderea C.I. de memorie în 1983 va fi de 48% la C.I. MOS şi de 1o% la C.I. bipolare [BERE83].

Evoluţia pe 1981, 1982 şi previziunile pentru 1983 pentru consumul de C.I. de memorie RAM sînt prezentate în fig.1.3 după datele publicate în [KOZM83].

In condiţiile evoluţiei rapide a complexităţii circuitelor integrate cît şi a sistemelor care sînt construite cu acestea, problema testării lor a devenit din ce în ce mai dificilă, costul testării reprezentînd un procent din ce în ce mai mare din preţul produsului finit [BATR81].

Studii asupra costului testării în fabricaţia C.I. de memorie au fost făcute de [BATR81], el tinzînd să depăşească 5o% în perioada 19oo-199o.

O trecere în revistă a problemelor testării UM cu semiconductoare a fost prezentată de autor în [DANC83a] şi [DANC83e].

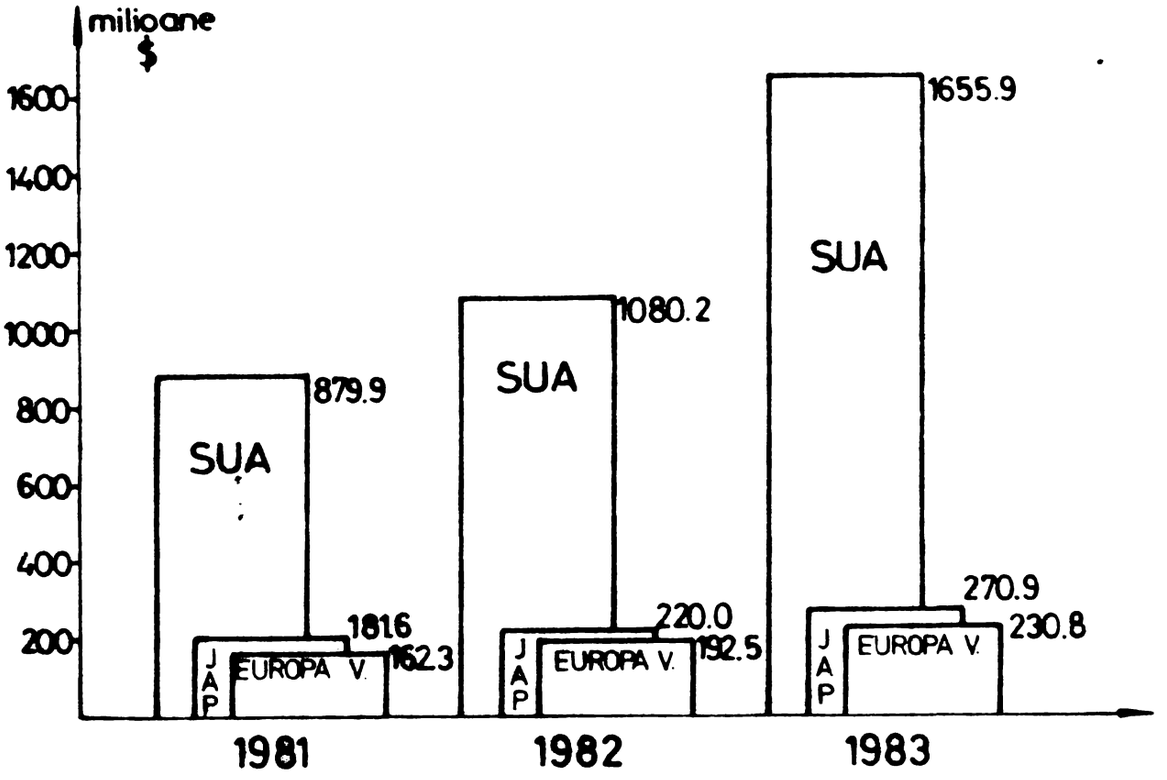


Fig.1.3. Consumul de C.I. de memorie RAM

Fig.1.3. Consumul de C.I. de memorie RAM.

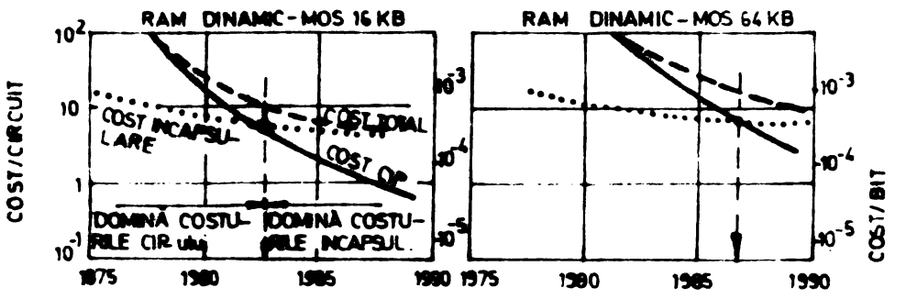


Fig.1.4. Perioadele de dominare a costurilor fabricației și testării C.I. de memorie MOS

Fig.1.4. Perioadele de dominare a costurilor fabricaţiei şi testării C.I. de memorie MOS.

Datorită presiunii industriei, ştiinţa testării are tendinţa de a se dezvolta ca o disciplină separată în dome-

nivul tehnicii de calcul, cu trei direcţii principale

- elaborarea secvenţelor de stimulare minime care să permită detectarea tuturor defectelor,
- automatizarea experimentului de testare,
- elaborarea de principii de proiectare a produselor în aşa fel încît ele să fie testabile.

Astfel evoluţia pieţii de echipamente de testare automată după datele publicate în [KOZM83] este prezentată în figura 1.5.

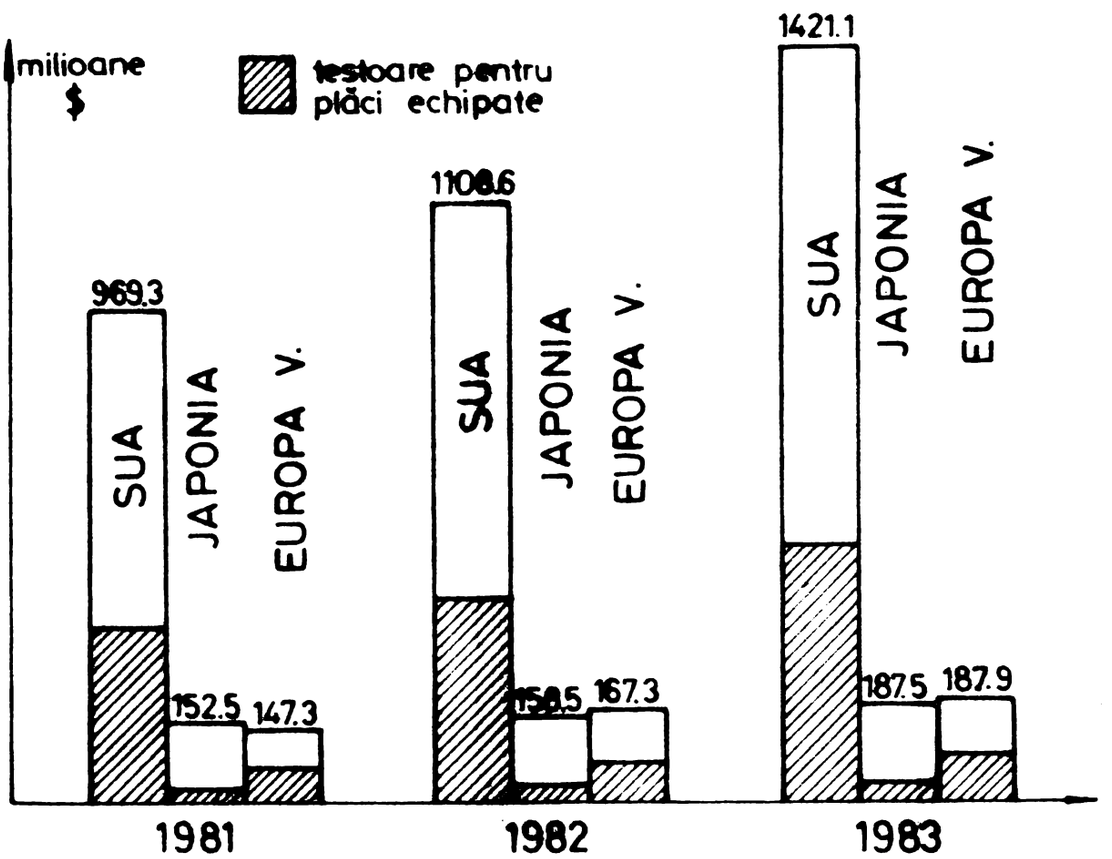


Fig.1.5. Evoluţia pieţei mondiale de echipamente de testare automată

In ţara noastră, realizări deosebite au fost obţinute la I.P.Cluj şi I.P.A. Cluj, prin lansarea în fabricaţie a familiei de echipamente de testare automată THETA ROM 5000. Această familie conţine testoare numerice de uz general (THETA 5010), testoare analogice (THETA 5020), pentru LSI (THETA 5030) şi pentru plăci hibride (THETA 5050) [SIRB82]. Trebuie menţionat de asemenea realizările obţinute la I.T.C. Timişoa-

ra (memorii), I.P. Timişoara (testarea plachetelor logice cu ajutorul funcţiilor de control prin sumă [VLAD82], IIRUC (testare de plachete), IAEM şi IP Timişoara (standuri de control automate).

In general se acceptă clasificarea metodelor de testare, după scopul lor în

testare < GO - NO/GO
testare < cu diagnoză (în vederea remedierii)

Fig.1.6.

In domeniul testării unităţilor de memorie, pe plan mondial cercetările sînt îndreptate în direcţia găsirii modelelor şi secvenţelor de test cît mai performante pentru testarea GO/NOGO şi a acelor secvenţe care să pună în evidenţă un singur defect (sau clasă de defecte), chiar şi în prezenţa altora, pentru identificarea şi remedierea lor.

In condiţiile actualei complexităţi a plăcilor cu circuite LSI, chiar cu un control interfazic corect implementat, în medie există un defect pe fiecare placă ajunsă la standul de test final [STON79]. Dintre acestea, cele mai frecvente sînt cele de scurtcircuit - 50% [PHIL80].

Testarea pentru defecte de scurtcircuit se poate executa pe testoare de continuitate tip "pat de cuie" (bed of nails), însă preţul acestor adaptoare (1 $ pentru un palpator) cu utilizare unică (pentru un singur tip de placă) şi fiabilitatea lor redusă, fac această metodă prohibitivă pentru plăci complexe, mai ales echipate. O soluţie ar putea fi constituită de un tip recent de testor de continuitate, cu sonde mobile [LYMA83].

Pe de altă parte, "pentru a micşora dificultăţile inerente ale testării şi a obţine maximum de randament prin testarea automată, este necesar ca testarea să fie tratată nu ca un eveniment singular ci ca un sistem, legat de întregul ciclu de viaţă al unui produs" [GREE82], începînd încă din faza de concepţie şi proiectare. Aşa cum se prezintă astăzi, testarea se concentrează asupra fazelor de producţie, control final şi recepţie la beneficiar.

Este necesară o informare şi educare a tuturor celor ce contribuie la realizarea de produse astfel încît acestea

să fie testabile, ducînd prin aceasta la ridicarea fiabilităţii lor precum şi la scăderea costurilor de fabricaţie.

*

* *

Lucrarea de faţă se integrează în domeniul preocupărilor crescînde din ţara noastră pentru testarea componentelor şi sistemelor în tehnica de calcul. Fără a neglija celelalte aspecte, autorul şi-a concentrat atenţia asupra elaborării de secvenţe de stimulare pentru diagnoza unităţilor de memorie cu C.I. şi asupra automatozării experimentului de testare.

Capitolul 2 prezintă o sinteză a defectelor circuitelor integrate de memorie RAM şi dificultăţile legate de testarea completă a memoriilor. Funcţionarea acestora este descrisă în anexa A.1 iar testele standard (test patterns) precum şi o comparaţie între ele sînt prezentate în anexa A.2.

Capitolul 3 ia în studiu problemele specifice ale testării cu diagnoză, pe echipamente de testare automată, a UM de memorie RAM şi strategiile ce pot fi adoptate pentru ridicarea fiabilităţii acestora cu un preţ de cost cît mai scăzut.

Capitolul 4 prezintă modalităţi de diagnosticare în sistem a UM cu C.I. Este generalizat un algoritm de diagnoză a liniilor de adresă şi este prezentat un algoritm de test original care identifică defectele de scurtcircuit între adrese. Performanţele acestor algoritmi sînt analizate printr-un sistem de 15 teoreme şi utilizarea unui aparat matematic original Sînt prezentate rezultatele obţinute prin simulare prin implementarea pe minicalculatorul I 100.

Capitolul 5, după o analiză a dezvoltării limbajelor de test, propune un nou limbaj de nivel înalt, MTL, pentru testarea automată a UM, implementabil pe ETA conduse de microcalculatoare de 8 biţi.

Sistemul software elaborat de autor pentru un astfel de ETA, alcătuit din sistem de operare pe disc flexibil, editor interactiv şi executiv pentru limbajul MTL este prezentat în capitolul 6. Anexa A.6 este alcătuită din listingul de asamblare al acestuia.

CAPITOLUL 2.

DEFECTE ALE C.I. DE MEMORIE RAM

2.1. Introducere

Problemele testării C.I. de memorie RAM şi ale unităţilor de memorie realizate cu acestea au stat în atenţia cercetătorilor încă de la începuturile fabricaţiei lor, cînd s-a constatat că ele sînt total diferite de cele ale testării memoriilor cu ferite.

Citînd surse bibliografice [BATR81] arată că defectele constatate la testarea finală a pastilelor de siliciu se datoresc în principal următoarelor cauze

- defecte prezente pe suprafaţa măştii fotografice, care cresc ca număr prin utilizarea repetată a măştii,
- defecte apărute pe suprafaţa de siliciu, datorită prelucrărilor ohmice sau termice, al căror număr creşte cu creşterea suprafeţei structurilor de siliciu şi cu micşorarea dimensiunilor elementelor de suprafaţă ale circuitului integrat (cu creşterea densităţii de integrare).

Odată cu creşterea densităţii de integrare şi creşterea dimensiunilor chipurilor, o problemă preponderentă a devenit numărul de defecte pe centimetru pătrat - în fond un indicativ al calităţii procesului tehnologic. Aceasta a scăzut de la 31 def./cm^2 la 5 def./cm^2 în perioada 1969-1979 [BATR81]. Chiar în aceste condiţii, probabilitatea de a obţine un randament de (15 ÷ 20)%, necesar rentabilităţii fabricaţiei de serie, este mică pentru C.I. de memorie de foarte mare capacitate; numai firma Texas Instruments raportează un randament de 50%, la fabricaţia C.I. de memorie de 64 Kb. Aceste probleme au dus la soluţia implementării unor linii şi coloane de rezervă şi reconfigurarea structurii prin reconfigurarea decodificatoarelor de adresă. Toate firmele producătoare de C.I. de memorie de 64 Kb (în afară de T.I) au recurs la această soluţie [POSA81b], [SUI81], [ABB081], [SMITH81], [GROS81]. Prezenţa simultană a unor C.I. cu arni-

tecturi diferite (prin reconfigurări diferite ale decodificatoarelor) va prezenta probleme de testare deosebite pentru UM, a căror soluţie nu este clară în momentul de faţă.

2.2. Defecte ale C.I. de memorie RAM

In general defectele sînt analizate pe baza unor modele de scheme logice ale C.I. Un grup de cercetători francezi [GALI78], [GALI8o] a pus în evidenţă existenţa unor defecte fizice fără corespondent în schemele logice, fără să dea însă o soluţie pentru testarea acestora ci doar evitarea apariţiei acestei clase prin respectarea anumitor reguli de proiectare. Rezultatele respective au fost însă aplicate la proiectarea unui microprocesor şi nu a memoriilor.

In cele ce urmează vor fi tratate numai defectele specifice memoriilor cu semiconductoare, şi nu se vor lua în considerare defectele circuitelor de intrare/ieşire; prin această clasă se înţeleg curenţii de intrare prea mari, fan-out mic, nivele de intrare sau ieşire necorespunzătoare, etc., defecte ce sînt puse în evidenţă de teste parametrice simple. Vor fi tratate numai acele defecte care duc la funcţionarea necorespunzătoare a C.I., privite ca memorii.

Aceste defecte pot fi grupate pe blocurile funcţionale ale memoriilor semiconductoare. Au fost propuse mai multe astfel de modele funcţionale [NAIR78], [FEE78], [TEKT74], [MONT75], [SUK81].

Se vor prezenta în continuare aceste defecte, luînd ca bază modelul elaborat de R.Nair [NAIR78]. El va fi completat cu modelul Suk [SUK81], precum şi cu defectele semnalate de alţi autori, deoarece [NAIR78] exclude acele defecte legate de comportarea dinamică a blocurilor funcţionale. [NAIR78] ia în considerare doar defecte ale matricii de celule de memorare, ale circuitelor de decodificare şi ale logicii de scriere/citire, incluzînd în aceasta din urmă toate celelalte elemente (linii de bit, amplificatoare de scriere, baza de timp internă etc.).

Modelele funcţionale prezentate, ca şi testele elaborate pe baza lor sînt pentru memorii cu lăţimea cuvîntului

de date de 1 bit, fiind uşor de extrapolat pentru cuvinte de mai mulţi biţi.

2.2.1. Defecte ale matricii de celule de memorare

Fiecare unitate adresabilă constă dintr-o singură celulă. Datorită unor cauze interne multiple, matricea de celule poate prezenta următoarele defecte [NAIR78]

1. Una sau mai multe celule blocate pe 0 sau 1.
2. Există una sau mai multe celule cuplate.

Prin aceasta înţelegem că, o tranziţie din starea x în starea $\bar{x}$ a unei celule i produce o tranziţie a unei alte celule j, din starea x în starea $\bar{x}$, sau din starea $\bar{x}$ în starea x. Acest lucru nu implică şi reciprocitatea, în sensul că o tranziţie în celula j nu produce o tranziţie în celula i. Vom denumi tranziţia unei celule din starea x în $\bar{x}$, care are loc în urma unei operaţii de scriere, ca tranziţie forţată.

Se pot defini astfel perechi ordonate de celule cuplate (i, j), în care celula j suferă o tranziţie datorată unei tranziţii forţate a celulei j. Pot exista N(N-1) perechi.

[NAIR78] prezintă o extindere a acestui model, pe care îl vom denumi modelul Nair extins.

3. Se defineşte o submulţime $S_{i,k}$ de k celule k - cuplate, în sensul că tranziţia forţată a unei celule din această submulţime produce tranziţia unei alte celule din $S_{i,k}$ din 1 în 0 sau din 0 în 1, dacă celelalte k-2 celule se află într-o anumită stare. Deoarece algoritmii de test ar ridica probleme deosebite pentru submulţimi arbitrare, [NAIR78] restrînge modelul de mai sus la submulţimi disjuncte, şi îl denumeşte modelul restrîns pentru k - cuplaje (restricted k - coupling model).

[SUK81] extinde acest model, pornind de la faptul că selecţia multiplă (defect al decodificatoarelor) nu este descrisă complet de modelul NAIR.

Modelul SUK ia în considerare următoarele defecte.

1. - Celule blocate pe 1 sau 0. Starea acestor celule va fi întotdeauna aceeaşi, indiferent de operaţiile de scriere sau citire efectuate asupra oricărei celule din matrice.

2. - Defecte de tranziţie: una sau mai multe celule nu efectuează tranziţii din 0 în 1 şi/sau din 1 în 0, cînd în celula respectivă se înscrie complementul conţinutului său.

3. - Defecte de cuplare: o tranziţie forţată într-o celulă poate induce tranziţii 0→1 sau 1→0 în alte celule din matrice, independent de conţinutul celorlalte celule (ca şi în modelul Nair). Acest lucru nu implică reciprocitatea.

Cele 3 defecte de mai sus, nu modelează corect [SUK81] defectele de selecţie multiplă. Atunci cînd mai mult de o celulă este selectată în cursul unei operaţii de CITIRE, rezultatul poate fi o funcţie SI sau SAU de conţinutul celulelor. De aceea SUK include încă o categorie de defecte.

4. - Defecte de acces multiplu: In cursul operaţiilor de scriere sau citire pot fi selectate mai mult decît o singură celulă. Dacă în cursul unei operaţii de CITIRE la anumite adrese sînt activate mai multe celule, rezultatul citirii este o funcţie SAU sau SI între conţinutul celulelor. Funcţia este întotdeauna aceeaşi pentru un C.I. dat, dar se presupune că ea nu este cunoscută a priori. In cursul operaţiilor de scriere, toate celulele activate trec în aceeaşi stare.

2.2.2. Defecte ale circuitelor de decodificare

Toate modelele elaborate restrîng setul de defecte la acea clasă care nu transformă circuitele de decodificare în maşini secvenţiale. Excluzînd această clasă, defectele se manifestă astfel [NAIR78]:

1. Decodificatorul nu va activa celula adresată. El poate în schimb activa alte celule.

2. Decodificatorul va activa mai multe celule, inclusiv cea adresată.

In cazul acceselor multiple nu se poate face distincţie între defecte ale decodificatoarelor sau ale matricii de memorare (cuplaje între celule). In cazul în care nu este activată nici o celulă, aceasta e văzută ca blocată pe 0 sau 1,

funcţie de logica utilizată. [NAIR78] demonstrează prin cele de mai sus că defectele decodificatoarelor se manifestă ca defecte ale matricii deci nu trebuie testate separat.

Mai sînt însă posibile defectele de comportare dinamică, neincluse în modelele [NAIR78] şi [SUK81].

[FEE78] pune în evidenţă şi alt tip de defect posibil, anume viteza acestor decodificatoare. O tranziţie mai lentă de la o adresă i la o adresă j se manifestă ca o mărire a timpului de acces la celula j. La memoriile fără registre de ieşire, variaţii ale timpului de acces cu adresa sînt observabile cu osciloscopul la pinul de ieşire. Apariţia memoriilor cu registre de ieşire, al căror tact este generat de o bază de timp internă, a complicat problema în loc să o simplifice. Creşterea timpului de acces se manifestă doar intern. Dacă intrarea registrului de ieşire nu este stabilă în momentul strobării, el reţine o altă dată decît cea citită intern din celula adresată. In general, timpul de acces la o celulă tinde să crească cu scăderea tensiunii de alimentare [FEE78] şi cu creşterea temperaturii. Dacă această bază de timp nu se modifică în acelaşi fel, memoria va putea apărea ca defectă în anumite condiţii de temperatură, deşi intern ea funcţionează corect.

2.2.3. Defecte ale logicii de scriere-citire

Unele linii ale amplificatoarelor de citire sau ale circuitelor de scriere pot fi blocate pe 1 sau 0. In ambele cazuri, aceste defecte pot fi incluse în clasa de defecte ale matricii de celule de memorare, considerînd celulele respective ca fiind blocate pe 1 sau 0 [NAIR78].

2.2.4. Condiţiile impuse unui test pentru a detecta defecte conform modelului Nair

Deoarece atît defectele decodificatoarelor cît şi cele ale logicii de scriere/citire se manifestă ca defecte ale matricii de celule de memorare, numai acestea din urmă trebuie testate în cadrul modelului Nair [NAIR78].

[NAIR78] enunţă 3 condiţii şi demonstrează că ele sînt necesare şi suficiente pentru a testa defectele din modelul propus

Condiţia 1 Fiecare celulă trebuie supusă:
- unei tranziţii forţate 0 - 1 şi
- unei tranziţii forţate 1 - 0

şi trebuie să fie citită înainte de a fi supusă unei alte tranziţii forţate.

Condiţia 2 Pentru orice pereche de celule (i, j), celula i trebuie citită după o tranziţie forţată în celula j şi înainte ca celulele i şi j să fie supuse altor tranziţii forţate, în următoarele cazuri

a) celula i în stare 0, celula j - tranziţie 0 → 1
b) celula i în stare 1, celula j - tranziţie 0 → 1
c) celula i în stare 0, celula j - tranziţie 1 → 0
d) celula i în stare 1, celula j - tranziţie 1 → 0

Condiţia 3 Pentru orice triplet de celule (i, j, k), dacă testul execută o tranziţie forţată a celulei j din y în $\bar{y}$ după o tranziţie a celulei i din x în $\bar{x}$ şi înaintea citirii celulei z, atunci trebuie să mai execute una din secvenţele de mai jos

a) celula k (în starea z) este citită după o tranziţie $x - \bar{x}$ în celula i şi înaintea unei tranziţii $y - \bar{y}$ în celula j, sau
b) celula k (în stare z) este citită după o tranziţie $y - \bar{y}$ în celula j şi înaintea unei tranziţii $x - \bar{x}$ în celula i.

Se demonstrează [NAIR78] că un test care respectă condiţiile de mai sus poate detecta orice perechi de celule cuplate.

Mulţi dintre algoritmii larg utilizaţi în testarea memoriilor semiconductoare (prezentaţi în anexa A2) nu au fost elaboraţi pe baza unui model sistematic de defecte ci pe baze intuitive [NAIR78]. Astfel, algoritmul MARCH nu satisface în general condiţiile 2 şi 3. Chiar algoritmi lungi, ca de pildă GALPAT, nu satisfac în întregime condiţia 2. Defectul, în care o tranziţie forţată 0 → 1 în celula i produce o tranziţie 1 → 0 în celula j, nu va fi detectat de GALPAT [NAIR78].

2.2.5. Defectul de pierdere a informaţiei la memoriile dinamice

Descărcarea capacităţii - celula de memorare - la acest tip de meorii, într-un interval de timp mai mic decît cel garantat de fabricant, reprezintă cauza acestei clase de defecte. Cauzele acestei descărcări pot fi multiple, iar timpul de descărcare este influenţat de

- temperatură
- tensiunea de alimentare
- informaţia conţinută în alte celule
- nivelul de radiaţii (în special radiaţii alfa).

Cercetări recente [BELL82] au arătat că, hidrogenul inclus în capsulele în care sînt împachetate C.I. de memorie, poate duce la defecte-datorită reacţiilor chimice iniţiate de radiaţia cosmică, sau de radiaţiile gazului Krypton ce rămîne inclus în capsulă.

Datorită cauzelor multiple, care pot duce la descărcarea acestei capacităţi şi a neuniformităţii caracteristicilor electrice şi dimensionale ale elementelor pe aceeaşi pastilă de siliciu, nu se poate defini o relaţie între timpul necesar între două regenerări ale acelciaşi celule şi temperatura ambiantă sau cea a joncţiunii.

Se acceptă în general o relaţie de forma

$$t_{ref} = A\, e^{-BT},$$

unde T - temperatura joncţiunii în ^{o}C

B - o variabilă ce leagă mărimea curentului de generare-recombinare de temperatura joncţiunii

A - o constanta ce depinde de aria celulei, amplificatoarele de citire etc.

Această relaţie ţine seama doar de una din cauzele de mai sus - creşterea curentului de scurgeri al capacităţii cu temperatura.

Valori tipice pentru B sînt între $0{,}053/^{o}C$ şi $0{,}080/^{o}C$ [OWEN79a] ceea ce se traduce prin reducerea la jumătate a timpului maxim între două regenerări la creşterea temperaturii cu $(11{,}6 \div 13{,}1)^{o}C$.

Alţi autori [KOPP76], [SUL81] situează la 10°C această valoare.

Testarea trebuie făcută la temperaturi ridicate, deoarece extrapolarea relaţiei de mai sus şi testarea la temperatura ambiantă (20 - 25°C) duce:fie la nedetectarea unor defecte care se manifestă doar la temperaturi ridicate,fie la eliminarea unor C.I. care-deşi prezintă erori la un timp de regenerare calculat pentru temperatura ambiantă-funcţionează corect la temperaturi înalte.

2.2.6. Defecte legate de informaţia înscrisă

Prezenţa defectelor, legate de informaţia înscrisă, sau a sensibilităţii la informaţie, a fost observată odată cu apariţia memoriilor semiconductoare. Această clasă de defecte se manifestă prin faptul că, pentru anumite configuraţii de informaţie, informaţia înscrisă în unele celule se alterează-fie în timp,fie în urma unor operaţii efectuate asupra altor celule sau asupra aceloraşi celule.

Sensibilitatea la informaţie a fost sesizată intuitiv, datorită posibilelor influenţe între celule. Nu a fost însă găsit pînă acum un model practic, cu ajutorul căruia această clasă de defecţiuni să fie studiată şi cu ajutorul căruia să poată fi definite teste practic executabile şi care să verifice toate defectele posibile.

In anul 1975, într-un articol de referinţă, J.P.Hayes a definit astfel un model, ca punct de plecare pentru testarea sensibilităţii la informaţie [HAYE75].

In cele ce urmează vom prezenta modelul şi algoritmul Hayes, deoarece acesta este un model universal valabil pentru memorii RAM şi prezintă amploarea problemelor ce se pun pentru o testare exhaustivă a acestor memorii.

2.3. Modelul şi algoritmul Hayes pentru testarea memoriilor RAM

2.3.1. Modelul memoriei Mr

O memorie Mr este definită ca o mulţime de r celule de

memorare C_0, C_1, C_i, C_{r-1}, indicele i fiind adresa celulei. Asupra fiecărei celule pot fi efectuate două tipuri de operaţii: SCRIERE şi CITIRE. Este convenabil să se presupună că o astfel de operaţie se poate referi la o singură celulă la un moment dat, adică lăţimea cuvîntului este de 1 bit. Organizarea memoriei Mr este prezentată în fig.2.1. După cum se poate observa este o simplificare a structurilor prezentate anterior, şi care priveşte memoria ca o cutie neagră.

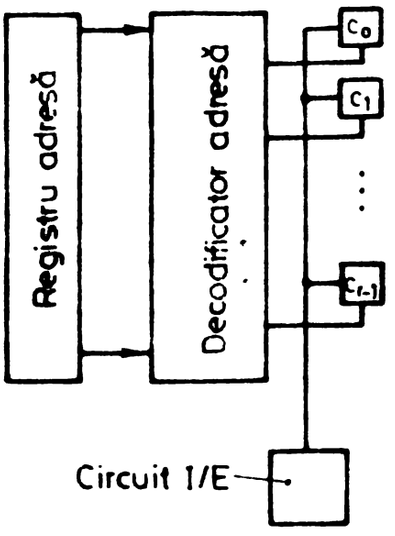


Fig. 2.1. Organizarea memoriei Mr

Fie $B=\{0,1\}$ şi B^r mulţimea tuturor celor 2^r vectori cu componente din B. Mr are $n=2^r$ stări distincte, fiecare stare $_j$ fiind un vector $(\mathcal{Y}_{j,0}, \mathcal{Y}_{j,1},\ldots,\mathcal{Y}_{j,r-1}) \in B^r$.

Asupra fiecărei celule C_i din Mr pot fi efectuate 3 operaţii distincte SCRIERE 1, SCRIERE 0 şi CITIRE. Aceste operaţii le vom denumi în continuare WRITE 1, WRITE 0 şi READ. Cu aceste operaţii asociem trei funcţii W_i, $\overline{W}_i$ şi R_i pe mulţimea stărilor, cu valori în mulţimea stărilor

$$W_i(\mathcal{Y}_{j,0},\mathcal{Y}_{j,1},\ldots,\mathcal{Y}_{j,i},\ldots,\mathcal{Y}_{j,r-1})=(\mathcal{Y}_{j,0},\mathcal{Y}_{j,1},\ldots,1,\ldots\mathcal{Y}_{j,r-1}) \quad (2.1)$$

$$\overline{W}_i(\mathcal{Y}_{j,0},\mathcal{Y}_{j,1},\ldots,\mathcal{Y}_{j,i},\ldots,\mathcal{Y}_{j,r-1})=(\mathcal{Y}_{j,0},\mathcal{Y}_{j,1},\ldots,0,\ldots\mathcal{Y}_{j,r-1}) \quad (2.2)$$

$$R_i(Y_j) = Y_j. \quad (2.3)$$

In cele ce urmează, se va nota cu $\tilde{W}_i$ o operaţie de scriere şi cu X_i orice operaţie.

Se poate defini funcţia de ieşire

$$Z(X_i,Y_j) = \begin{cases} \mathcal{Y}_{j,i} & \text{dacă } X_i=R_i \\ " - " & \text{dacă } X_i=\tilde{W}_i \end{cases} \quad (2.4)$$

Pe baza notaţiilor de mai sus, o memorie Mr poate fi considerată ca un automat Mealy avînd :

$n=2^r$ stări

3r simboluri de intrare W_i, $\overline{W}_i$, R_i, i=o,1,...,r-1

2 simboluri de ieşire, cu funcţia de ieşire definită de (2.4).

Tranziţiile între stări sînt date de relaţiile (2.1) - (2.3).

Tabelul 2.1 conţine stările unei memorii cu 2 celule, M2 iar în fig.2.2 este prezentat graful tranziţiilor.

Tabelul 2.1 Stările memoriei M2.

Starea iniţială	INTRARI					
	W_o	W_1	$\overline{W}_o$	$\overline{W}_1$	R_o	R_1
00	10,-	01,-	00,-	00,-	00,0	00,0
01	11,-	01,-	01,-	00,-	01,0	01,1
10	10,-	11,-	00,-	10,-	10,1	10,0
11	11,-	11,-	01,-	10,-	11,1	11,1

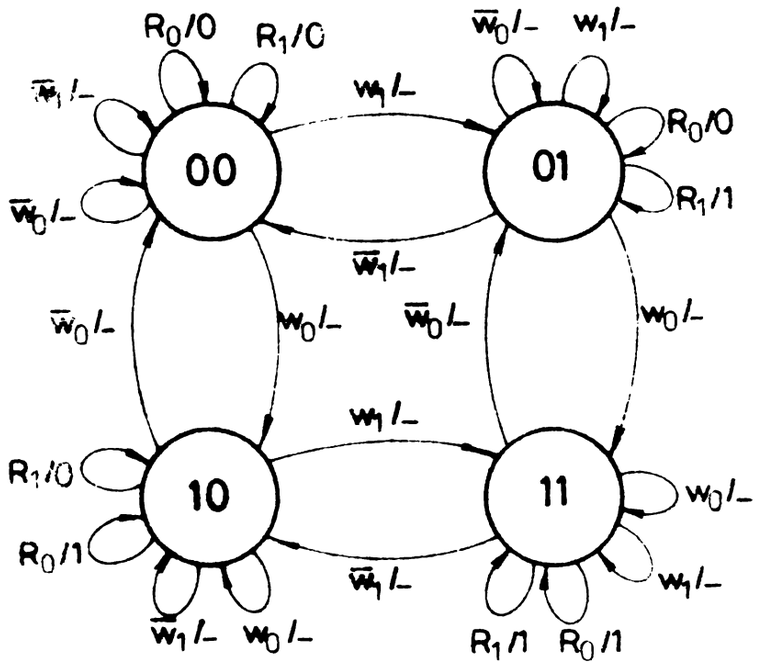


Fig.2.2.G2 - Graful tranziţiilor pentru M2

Comportarea memoriei Mr este complet descrisă de mulţimea funcţiilor $F_o=\{ W_i, \overline{W}_i, R_i, Z\}$, unde i=o,1,...,r-1. Un defect de sensibilitate la informaţie (Pattern sensitive fault, PSF), F există dacă funcţiile $\{W_i, \overline{W}_i, R_i, Z\}$ se mo-

difică în $\{W_i^F, \bar{W}_i^F, R_i^F, Z^F\}$, unde X_i^F pot fi orice aplicaţii definite pe B^r cu valori în B^r, iar Z^F este o aplicaţie oarecare definită pe $(B^r)^2$ cu valori în B.

Vom nota defectul, (simplu sau multiplu), ca mulţimea funcţiilor

$$F = \left\{ W_i^F, \bar{W}_i^F, R_i^F, Z^F \right\}$$

iar maşina secvenţială în care se transformă Mr în prezenţa defectului F, cu Mr^F.

Sînt necesare cîteva precizări în legătură cu stările memoriei Mr^F:

a. Starea internă - (Internal state), este configuraţia de 1 şi 0 efectiv înscrisă în celulele $C_0 \ldots C_{r-1}$, şi este un vector din B_r

$$Y_j = (y_{j,o}, y_{j,1}, \ldots, y_{j,r-1})$$

b. Stare aşteptată - (Expected state). Fie S o secvenţă de sincronizare, a cărei aplicare la intrarea automatului Mr fără defect îl duce într-o stare Y_k, independent de starea iniţială (orice secvenţă care conţine o scriere în fiecare celulă are această proprietate). Se aplică aceeaşi secvenţă S memoriei Mr^F cu defect, şi fie Y_j starea memoriei Mr^F după aplicarea secvenţei de sincronizare. Această stare poate chiar să fie dependentă de starea iniţială a memoriei defecte, deci secvenţa S să nu fie secvenţă de sincronizare pentru Mr^F. Starea aşteptată a memoriei Mr^F este starea Y_k, în care ar fi ajuns o memorie fără defect după aplicarea secvenţei de sincronizare S. Este convenabil să considerăm starea aşteptată a memoriei Mr^F ca o funcţie de starea sa internă

$$E(Y_j) = Y_k .$$

c. Starea aparentă - (Apparent state), este acea stare a memoriei Mr^F care poate fi dedusă în urma aplicării unei secvenţe de distingere. In

cazul memoriilor, o secvenţă care citeşte toate celulele este o secvenţă de distingere, deci starea aparentă poate fi definită ca

$(Z^F(R_o,Y_j), Z^F(R_1, Y_j),...,Z^F(R_{r-1}, Y_j)$ şi se notează ca $A(Y_j)$.

Dacă Mr^F=Mr toate cele trei stări coincid.

Un defect de sensibilitate la informaţie este <u>detectabil</u> dacă şi numai dacă $\exists$ o stare internă Y_j pentru care, la un moment dat, starea aşteptată nu coincide cu starea aparentă $A(Y_j)=E(Y_j)$.

Un PSF este nedetectabil, dacă starea internă diferă de cea aşteptată, dar starea aparentă coincide întotdeauna cu cea aşteptată. Există o corespondenţă 1 la 1 între defectele de sensibilitate la informaţie nedetectabile şi permutările în B^r.

Se pune deci problema testării memoriei Mr pentru PSF detectabile. Un defect F poate schimba maşina secvenţială Mr, într-o altă maşină secvenţială Mr^F, cu $n=2^r$ sau mai puţine stări. Hayes deduce un algoritm, (secvenţă de verificare), care să permită deosebirea maşinii secvenţiale Mr de orice altă maşină secvenţială cu maximum n stări, cu care este incompatibilă.

Memoria Mr, ca maşină secvenţială, are unele proprietăţi, care facilitează deducerea secvenţei de verificare:

1. Orice secvenţă, de r citiri distincte, identifică starea iniţială a memoriei Mr şi de aceea constituie o secvenţă de distingere pentru Mr.

2. Memoria Mr poate fi trecută în orice altă stare, printr-o secvenţă de maximum r operaţii WRITE asupra unor celule distincte. Mr are, deci, un set complet de secvenţe de sincronizare.

3. Graful tranziţiilor este un graf eulerian. In fiecare nod intră 3r arce, (r arce de citire şi 2r arce de scriere) şi din fiecare nod pornesc 3r arce (r citiri, r scrieri care nu modifică starea, r scrieri care modifică starea).

Fiecare nod are 2r bucle locale, adică încep şi se sfîrşesc în acelaşi nod. Fie Hr un subgraf al lui Gr, obţinut din acesta prin eliminarea buclelor locale. Subgraful Hr este un graf eulerian. H_2 pentru M2 este reprezentat în fig.2.3.

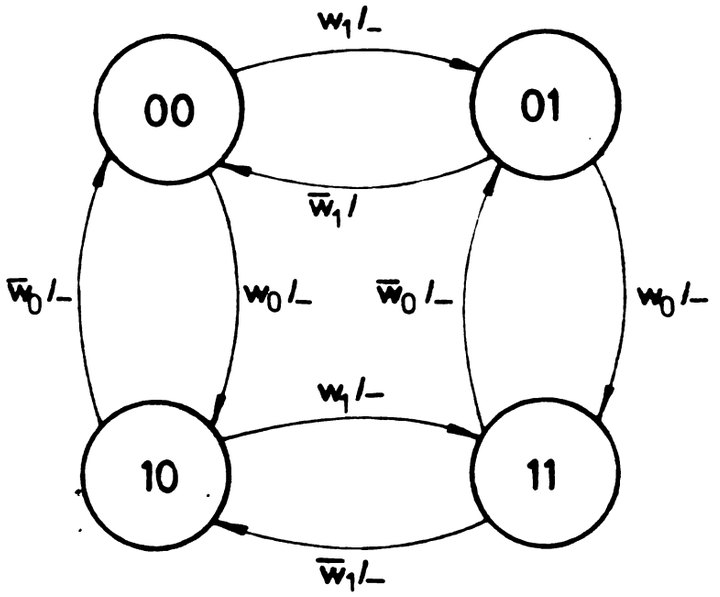


Fig. 2.3. Subgraful H2

2.3.2. Construirea secvenţei de verificare

Un drum Eulerian în Hr corespunde cu o secvenţă S de operaţii WRITE distincte, care schimbă starea memoriei Mr. Dacă fiecare element al secvenţei S este urmat de o secvenţă de distingere

$$D_o = R_o\ R_1 \ \dots\ R_{r-1}\ ,$$

atunci există posibilitatea de a verifica fiecare arc din Hr, presupunînd că operaţiile READ sînt corecte.

Fiecare buclă locală W_i din G_r poate fi verificată în acelaşi mod.

In final, trebuie verificate toate tranziţiile READ asociate fiecărei stări.

Fie D_i o secvenţă de distingere, obţinută din D_o prin rotirea cu i poziţii la stînga

$$D_i = R_i R_{i+1} \dots R_{r-1} R_o \dots R_{i-2} R_{i-1}\ .$$

Fie $D = D_o D_1 \ \dots\ D_{r-1}$.

Hayes demonstrează [HAYE75] că secvenţa $D_o D$ aplicată memoriei Mr în fiecare stare verifică toate tranziţiile READ.

Pasul 1.

Se construieşte o secvenţă eulerianã de operaţii WRITE din Hr, începînd din starea Y_0.

$$E = V_1 V_2 \dots V_{rn} ,$$

unde $V_i \in \{\tilde{W}_0, \tilde{W}_1, \dots, \tilde{W}_{r-1}\}$, pentru $1 \leq i \leq rn$.

Pasul 2.

După fiecare operaţie din E se înserează o secvenţă de distingere D_0

$$T = V_1 D_0 \; V_2 D_0 \dots V_{rn} D_0 .$$

Pasul 3.

Fie $Y_i=(y_{i,0}, y_{i,1}, \dots, y_{i,r-1})$ o stare din Mr.

Fie $W'_j=W_j$ dacă $y_{i,j}=1$ şi $W'_j=\bar{W}_j$ dacă $y_{i,j}=0$ (W'_j este o scriere în celula j, care nu modifică starea celulei).

Definim

$$L_i = W'_0 D_0 W'_1 D_0 \dots W'_{r-1} D_0 D, \qquad 0 \leq i \leq n-1 .$$

Secvenţa L_i verifică toate arcele locale asociate stării Y_i.

Se înserează secvenţa L_i după fiecare secvenţă $V_j D_0$ din T. Fie T' secvenţa rezultată.

Pasul 4.

Fie S_0 o secvenţă de sincronizare care duce memoria în starea Y_0. Se precede secvenţa T' cu secvenţa de sincronizare şi distingere $S_0 D_0$.

Secvenţa de verificare propusă este

$$C_r = S_0 D_0 T' .$$

O astfel de secvenţă are lungimea de $(3r^2+2r)2^r$.

Presupunînd că un ciclu de memorie este de 500 ns, testarea unei memorii de 16 K cuvinte ar dura:

$$10^{4927.186} \text{ ani} .$$

Deşi algoritmul prezentat este "aproape optimal" [HAYE75], pentru o verificare exhaustivă a memoriei Mr-privită ca maşină secvenţială-el nu este practic implementabil.

Din acest motiv, trebuie restrîns foarte mult setul de defecte de sensibilitate la informaţie, pentru care se face testarea.

O cale, de reducere a lungimii acestei secvenţe de verificare, este restrîngerea testului-în aşa fel încît să detecte-

ze numai sensibilităţi locale. Inţelegem prin aceasta restrîngerea setului de defecte pentru care se face testarea : defectul, pentru o celulă C_i, poate fi datorat numai interacţiunilor dintre celulele dintr-o vecinătate N_i a acestei celule, astfel încît $C_i \in N_i$. Astfel, celulele din N_i pot fi denumite "adiacente". Această adiacenţă poate fi determinată de mai multe fenomene:

1. Adresele celulelor din N_i pot fi într-o relaţie dată. Una din sursele posibile de eroare o constituie fenomenele de cursă sau hazard în circuitele de decodificare, şi astfel o celulă C_j este adresată, în cursul unei operaţii de scriere sau citire, în locul celulei C_i.

2. Celulele sînt fizic adiacente, sau liniile lor de intrare/ieşire sînt adiacente.

Interacţiunile electromagnetice dintre aceste linii se manifestă ca interacţiuni între diverse celule din N_i.

Fiind dat un set complet de vecinătăţi

$N = \{ N_0, N_1, \ldots, N_{r-1} \}$ pentru Mr, un defect local în raport cu această partiţionare are următoarele proprietăţi:

1. Funcţiile X_i^F, Z^F sînt independente de conţinutul celulelor din afara vecinătăţilor date $C_j \notin N_i$.

2. Operaţiile de scriere/citire X_i^F asupra celulei C_i nu pot altera conţinutul celulelor din afara vecinătăţii N_i.

N se numeşte un <u>set închis</u> de vecinătăţi dacă $\forall N_i \in N$, $C_j \in N_i \Rightarrow N_j = N_i$. Cu alte cuvinte vecinătăţile sînt disjuncte. Un set de vecinătăţi care nu este închis se numeşte <u>deschis.</u>

Pentru seturi închise de vecinătăţi se pot defini teste mai scurte, care constă în secvenţe de verificare, de tipul celor descrise anterior, pentru fiecare vecinătate.

Lungimea acestei secvenţe va fi $p(3q^2+2q)?^2$, unde q este numărul de celule dintr-o vecinătate, iar p numărul vecinătăţilor, astfel încît r=pq.

Nu se poate partiţiona o memorie semiconductoare într-un set închis de vecinătăţi, care să conţină un număr convenabil de mic de celule, deoarece nu putem exclude interacţiunile dintre celule fizic adiacente şi care ar fi incluse arbitrar în vecinătăţi disjuncte.

Au fost elaborate modele şi încercări de partiţionare ale memoriei în vecinătăţi [SUKOo], [HAYE8o], [SUHO1], însă

fără rezultate notabile, deoarece partiţionarea nu poate fi făcută decît în strînsă legătură cu structura şi topologia internă, specifică fiecărui C.I. de memorie în parte.

CAPITOLUL 3.

TESTAREA CU DIAGNOZĂ A UNITATILOR DE MEMORIE RAM CU C.I.. TEHNICI SI STRATEGII

3.1. Structura generală a unei unităţi de memorie

Numărul de cuvinte dintr-o unitate de memorie depăşeşte în general numărul de cuvinte dintr-un circuit integrat de memorie. Din acest motiv, unităţile de memorie utilizează frecvent un aranjament de circuite integrate de memorie sub formă de matrice. Considerînd circuitele integrate ca avînd o organizare de Nxl bit, pentru o memorie de q.N cuvinte a W biţi se va utiliza o matrice de qxW circuite integrate de memorie. Atît q cît şi N sînt în mod uzual puteri ale lui 2.

Intr-o astfel de matrice, fiecare coloană reprezintă o memorie de qN.1 bit. Prin urmare, în cazul unei operaţii de scriere/citire asupra unităţii de memorare, aceasta execută operaţia respectivă asupra tuturor circuitelor integrate de memorie dintr-un singur rînd. Toate intrările de date şi toate ieşirile de date ale circuitelor dintr-o coloană sînt legate între ele, pe principiul magistralei.

Structura generală a unei astfel de unităţi de memorare este dată de fig.3.1.

In afară de liniile de date, toate celelalte linii din matricea de memorare sînt orizontale.

Deoarece am considerat matricea de celule din capsula de memorie ca avînd două axe x şi y, vom introduce o a 3-a axă z, adresa z este numărul rîndului de circuite de memorie din unitate. Vom denumi în continuare circuitele de decodificare ca decodificatoare pe axa z, pentru a le distinge de decodificatoarele aflate în interiorul C.I. de memorie (de rînd (pe x), de coloană (pe y)).

Unul sau mai multe din blocurile reprezentate cu linie subţire pot lipsi în cazul unităţilor mai simple.

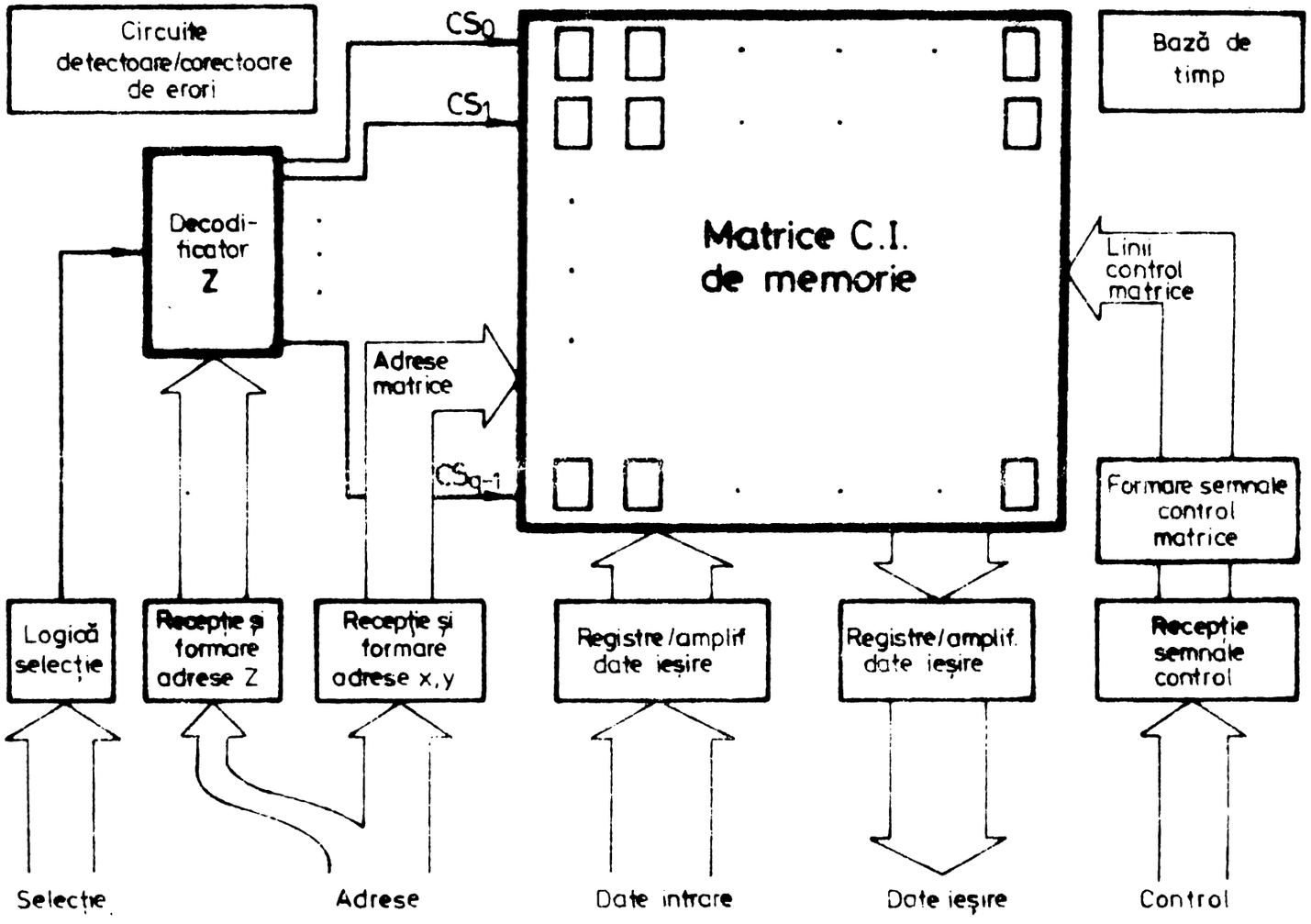


Fig. 3.1. Structura unei unități de memorie

3.2. Model de defecte pentru unităţi de memorie cu semiconductoare

Pentru a putea generaliza modelul de eroare pentru toate tipurile de unităţi de memorie care respectă schema bloc din fig.3.1 se va face abstracţie de restul blocurilor şi se vor considera doar acele defecte care se traduc prin semnale necorespunzătoare pe liniile din matrice [CHAS77a], [SRIN78].

Aceste linii sînt de următoarele tipuri:

- Linii de date leagă între ele toate intrările (ieşirile de date ale C.I. dintr-o coloană;
- Linii de adresă leagă între ele toate C.I. din matrice. In unele cazuri există emiţătoare de adrese separat pe fiecare rînd;
- Linii de control scriere, citire, regenerare, CAS, etc.;
- Linii de selecţie CS0, CS1,...,CS_{q-1} ale C.I. dintr-un rînd.

Defectele unei unităţi de memorare pot fi clasificate astfel

- defecte ale C.I. de memorie RAM
- defecte ale liniilor -de date,
  -de adrese,
  -de selecţie,
  -de control

Prin defect al unei linii se înţelege:

- blocată pe 1/0.
- în scurtcircuit cu altă linie din matrice.
- necorespunzătoare datorită unui defect în blocul de generare a semnalului.

Testarea funcţională tip GO-NO GO a unităţii de memorie nu ridică probleme deosebite faţă de testarea C.I. de memorie, fiind necesare în plus doar cîteva teste care să detecteze scurtcircuite sau cuplaje între liniile de date. Aceste teste vor fi analizate mai jos.

In cele ce urmează se va trata problema localizării defectelor la unităţile de memorie RAM cu semiconductoare, avîndu-se în vedere testarea cu depanare a acestor unităţi.

La plăcile de memorie RAM complexe utilizate azi în tehnica de calcul, există în medie cel puţin un defect pe placă atunci cînd placa ajunge la standul de test final complet echipată, chiar dacă se iau măsuri de testare interfazică şi de testare a componentelor înainte de montaj [STON79].

Cele mai frecvente defecte sînt [PHIL8o]:

- linii în scurtcircuit 5o%
- C.I. defecte 2o%
- cablaje defecte 2o%
- C.I.(logice) lente 2%
- linii întrerupte 2%
- componente neimplantate 2%
- componente (C.I.) plantate greşit 2%

3.3. Harta erorilor pe modul - tehnică modernă de diagnoză

Dacă pentru plăcile uzuale cu circuite logice se poate face o testare tip GO-NO GO "in vivo" adică chiar în sistemul în care placa urmează să funcţioneze, testarea plăcilor de memorie "in vivo", prin program, nu este satisfăcătoare deoarece testul nu se execută la viteza de lucru a memoriei. Aceasta se datorează faptului că între ciclurile de acţionare a memoriei testate, unitatea centrală execută citirea şi decodificarea instrucţiilor din programul de test, care nu este rezident în memoria testată.

De la început au fost utilizate testoare specializate care acţionau memoria după unul sau mai mulţi din algoritmii prezentaţi anterior. Tehnica utilizată în aceste sisteme - oprire la eroare şi afişarea adresei, datelor citite şi a celor aşteptate - era nesatisfăcătoare pentru depanare deoarece indica o eroare şi nu un defect. De pildă defectul "C.I. de memorie cu ieşirea blocată pe 1" se manifestă la testul MARCH prin 2N erori. In cazul utilizării unei capsule de 4K, ar fi necesare 8192 opriri şi interpretări ale datelor afişate.

Din acest motiv au fost dezvoltate tehnici noi, cum este de pildă Harta erorilor pe modul. Aceasta este denumită Board Error Map (BEM) de [CHA377a], [CHA377b] sau Board Array

Map [MACR78]. Aceasta constă într-o reprezentare simbolică în plan a matricii de C.I. de memorie, cu un simbol pentru C.I. la citirea cărora au fost detectate erori şi alt simbol pentru C.I. fără erori (fig.3.2).

Esenţial este faptul că alcătuirea acestei hărţi se face în timp real pe durata testului, permiţînd acţionarea memoriei la viteza de lucru.

După terminarea testului/testelor ea este afişată pentru interpretare.

Afişarea se poate face prin diode LED sau pe un display /imprimantă în cazul testelor controlate de mini/macrocalculatoare.

rînd	nr.bit date								
	0	1	2	3	4	5	6	7	8
A	.	.	.	E	.	.	.	.	.
B	.	.	.	.	E	.	.	.	.
C	.	.	.	.	E	.	.	.	.
D	.	.	.	.	.	.	.	.	.

Fig.3.2. BEM pentru o memorie de 9 biţi cu 4 rînduri de C.I. de memorie.

3.4. Probleme legate de testarea pentru depanare a unităţilor de memorie cu C.I.

Diagnosticarea unităţilor de memorie cu C.I. este în general o problemă dificilă din următoarele motive [CHAS77a], [CHAS77b], [PHIL80]:

1. Unităţile de memorie sînt formate din cel puţin 3 nivele logice

- amplificatoare/registre pentru date de intrare (TTL)
- C.I. de memorie (MOS)
- amplificatoare/registre pentru date de ieşire (TTL)

De pildă un defect pe un bit de date poate fi cauzat de oricare din elementele de mai sus.

2. Unele defecte pot masca alte defecte.

Exemplu 1. Un defect pe o linie de adresă din matrice face ca întreaga memorie să apară ca defectă (nefiind îndeplinirea condiţiei de unicitate a adresei pe toţi biţii de date). Acest defect poate masca un defect pe un bit de date (C.I. de memo-

rie defect).

3. Memoriile prezintă sensibilităţi la modificarea tensiunilor de alimentare, a relaţiilor de timp între semnalele de intrare, la temperatură precum şi sensibilitate la informaţie. Acest lucru implică faptul că fiecare placă, chiar fără defecte de fabricaţie trebuie supusă unei testări riguroase care să garanteze funcţionarea ei în orice condiţii de funcţionare (în limita specificaţiilor tehnice). In plus, chiar dacă se utilizează C.I. de memorie testate, la introducerea lor în matrice apar noi sensibilităţi datorită cuplajelor între liniile de semnal şi/sau zgomotelor de pe placă.

3.5. Identificarea defectelor unităţilor de memorie

In cele ce urmează ne vom referi doar la defectele enumerate la începutul acestui capitol, adică defecte pe liniile din matricea de circuite de memorie.

Pentru a elimina mascarea unor defecte de alte defecte au fost dezvoltaţi algoritmi de test specializaţi în detectare defectelor liniilor de date, CS, adrese şi control, astfel încît (pe cît posibil) defecte ale unor linii (sau C.I. de memorie) să nu mascheze alte defecte [CHAS77a], [CHAS77b], [SRIN78]. Interpretarea rezultatelor se face prin BEM, fie de către operator fie de către un calculator.

Intotdeauna testarea se începe prin verificarea liniilor de control, CS, date şi adrese, în această ordine pentru a evita mascarea defectelor.

3.5.1. Verificarea liniilor de control

Asupra unei singure locaţii din fiecare rînd de C.I. de memorie se execută secvenţa

- se înscrie 0 şi se citeşte 0
- se înscrie 1 şi se citeşte 1

Dacă există cel puţin un bit dintr-un rînd în care nu s-a detectat o eroare înseamnă că liniile de control funcţionează corect. Defecte ale liniilor de adresă nu maschează de-

fecte ale liniilor de control deoarece înscrierea şi citirea se fac din aceeaşi celulă, dacă se presupune că ieşirea blocului formator de adrese este întotdeauna aceeaşi pentru acelaşi cuvînt de adresă aplicat la intrare.

3.5.2. Verificarea liniilor CS şi date

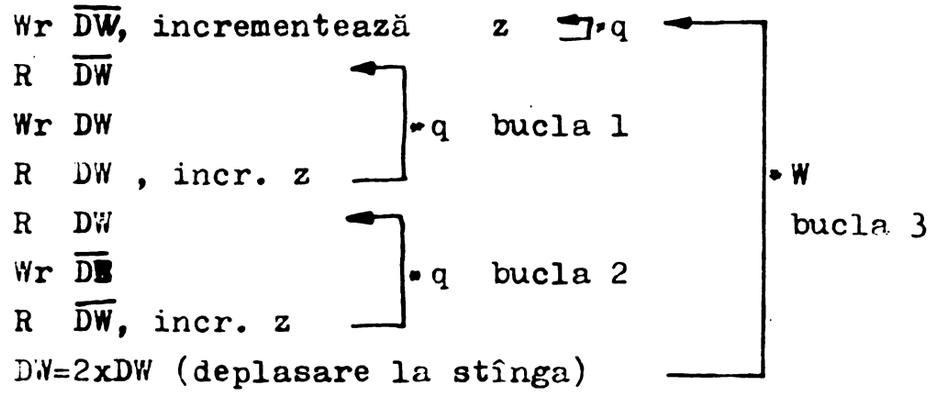
Teste pentru verificarea liniilor CS şi de date au fost dezvoltate de [CHAS77a], [CHAS77b] şi [SRIN78]. Se va prezenta în continuare testul EXISTENTA [CHAS77a], [CHAS77b] modificat.

Pasul 1.

Se iniţializează toate locaţiile memoriei pe 00...0.

Pasul 2.

Cu x=y=0 (pentru a evita mascarea de către defecte de adresă) se execută o secvenţă de MARCH pe axa z, folosind cuvîntul de date DW=00...01



q= nr.de rînduri pe axa z

W= lungime cuvînt

Fig.3.3. Testul EXISTENCE.

Analiza defectelor

a. Linie de date blocată

- acest defect va fi detectat în fiecare trecere a fiecărei bucle, deoarece fiecare buclă conţine atît R DW cît şi R $\overline{DW}$.

b. Scurtcircuit între două căi de date.

Pentru orice pereche de linii b_j şi b_i există o buclă în care informaţia pe aceste linii este complementară.

Această condiţie nu este îndeplinită de testul enunţat de [CHAS77a], [CHAS77b], care nu utilizează bucla 3 şi care

utilizează DW = 10101010...

c. Scurtcircuit între liniile de date de intrare şi ieşire în aceeaşi coloană.

Testul EXISTANCE modificat ca mai sus poate detecta şi acest tip de defect care nu este tratat în literatură.

Defectul este relativ frecvent, cele două linii fiind de obicei alăturate. Condiţia ca el să fie detectat este ca pe durata unei ciclu de citire pe linia de date de intrare din matrice să existe complementul stării aşteptate pe linia de date de icşire din matrice. Această condiţie nu este în general satisfăcută.

d. Defecte ale liniilor CS.

In BEM rîndurile al căror CS este necorespunzător vor fi defecte în întregime.

In cele ce urmează se vor prezenta formele în care apar în BEM defectele de mai sus.

	8	7	6	5	4	3	2	1	o
z=00 A	.	.	.	E	.	E	E	.	.
z=01 B	.	E	.	E	.	E	E	.	.
z=10 C	.	.	.	E	.	E	E	.	.
z=11 D	.	.	.	E	.	E	E	.	.

Fig.3.4.

In fig.3.4 este prezentată BEM pentru defectul multiplu:
1.Linia de date 5 defectă
2.Liniile de date 2 şi 3 în scurtcircuit.
3.C.I. 7B defect.

	8	7	6	5	4	3	2	1	o
z=00 A	.	.	.	E	.	.	.	.	.
z=10 B	E	E	E	E	E	E	E	E	E
z=10 C	.	.	.	E	.	.	.	.	.
z=11 D	.	.	.	E	.	.	.	.	.

Fig.3.5.

In fig.3.5 este prezentată BEM pentru defectul multiplu:
1.Linia CS1 defectă
2.Linia de date 5 defectă

După cum se poate observa, cele două defecte pot fi identificate.

	8	7	6	5	4	3	2	1	o
z=00 A	.	.	.	E	.	.	.	.	.
z=01 B	E	E	E	E	E	E	E	E	E
z=10 C	.	.	.	E	.	.	.	.	.
z=11 D	E	E	E	E	E	E	E	E	E

Fig.3.6.

In fig.3.6 este prezentată BEM pentru defectul multiplu:
1.Linia de date 5 defectă
2.Linia A_{zo} blocată pe 0 (înainte de decodificatorul pe z).

Fig.3.7.

In fig.3.7 este prezentată BEM pentru defectul multiplu:

1. Linia de date 5 defectă.
2. Liniile A_{zo} şi A_{z1} în scurtcircuit.

3.5.3. Verificarea liniilor de adresă din matricea de C.I.

Dacă a este numărul de biţi de adresă ai fiecărui C.I. de memorie (x şi y) astfel încît $N=2^a$, fie P mulţimea adreselor care au un singur bit pe 1

$$P = \{ A_j \mid A_j=2^j,\ j=0,1,\ldots,a-1 \}$$

şi fie $\bar{P}$ mulţimea adreselor care au un singur bit pe 0

$$\bar{P} = \{ B_j \mid B_j=\text{not } 2^j,\ j=0,1,\ldots,a-1 \}$$

Acest test se execută pe fiecare submulţime de C.I. care are amplificatoare şi linii de adresă separate. Se poate executa testînd un singur bit, pe ale cărui linii de date nu au fost detectate defecte cu testul de existenţă.

In legătură cu mulţimea locaţiilor pe care se desfăşoară testul sînt valabile teoremele T1÷T9 din capitolul 4. In acelaşi capitol sînt analizaţi alţi doi algoritmi [SRIN78], [DANC83b] pentru verificarea liniilor de adresă.

[CHAS77b] indică utilizarea următoarei secvenţe pentru testul de adrese.

Pasul 0 iniţializarea pe 0 a tuturor celulelor
1 Wr 0 în celula C_o
2 Wr 1 în celula C_{Aj}
3 R 1 din celula C_{Aj}
4 R 0 din celula C_o
END

Prin repetarea secvenţei de mai sus pentru fiecare rang de adresă (8=o,1,...,a-1) se pot identifica toate adresele defecte.

Analiza defectelor.

a. Linia de adresă j, blocată pe 0.

In acest caz în pasul 2 se va scrie 1 în celula 0 iar în pasul 4 citirea celulei va detecta eroarea.

b. Linia de adresă j blocată pe 1.

In acest caz în pasul 4 se va citi 1 din celula C_{Aj} în loc de 0 din celula C_o şi se va detecta o eroare.

c. Linii de adresă în scurtcircuit.

Fie linia j în scurtcircuit cu linia k. Dacă amplificatoarele de adresă sînt TTL, atunci, cele două linii realizează funcţia logică SI cablat.

In pasul 2 şi 3 scrierea şi citirea se va face tot în celula C_o (linia K fiind 0, şi linia j va fi tot 0 datorită funcţiei SI CABLAT).

Deci în pasul 4 celula C_o va conţine 1 în loc de 0.

Testul nu detectează scurtcircuite între linii de adresă dacă acestea realizează funcţia SAU cablat. Aceste defecte sînt detectate de testul complementat în care se citesc celulele C_{N-1} şi $C_{Bj} \in P$.

Testînd în modul de mai sus toate liniile de adresă din toate rîndurile, rezultatele se pot exprima prin diagrama adreselor defecte (Address Fault Plot - AFP).

Bit	o	1	2	3	4	5	6	7	8	9	1o	11
Rînd A	.	.	F	F	.	.	.	.	F	.	.	.
Rînd B	.	.	F	F	.	.	F	.	F	.	.	.
Rînd C	.	.	F	F	.	.	.	.	F	.	.	.
Rînd D	.	.	F	F	.	.	.	.	F	.	.	.

Fig.3.8. Diagrama adreselor defecte.

Fig.3.8 reprezintă AFP pentru un modul de memorie de 16K cuvinte (4 rînduri de C.I. de 4K).

Sînt indicate ca defecte liniile de adresă 2,3 şi 8.

In cazul în care capsulele sînt cu adrese multiplexate - adică pentru adresele de x şi y de acelaşi rang există fizic o singură linie, un defect pe această linie trebuie să se manifeste atît pe adresa de x cît şi pe cea de y (2 şi 8). Linia de adresă 3 din matrice este cu siguranţă fără defect

deoarece un defect pe această linie ar fi trebuit să fie detectat şi de secvenţa pentru testarea liniei 9. Defectul se găseşte prin urmare înainte de/sau în multiplexorul de adrese.

V.P.Srini [SRIN78] a prezentat în anul 1978 un test cu aceleaşi funcţii dar la care cuvîntul de date înscris la fiecare din locaţiile $A_j \in P$ este numărul rangului de adresă incrementat cu unu (j+1) codificat cu un cod corector de erori.

Dacă la citirea cuvîntului cu A_j, după corecţia erorilor nu se regăseşte informaţia j+1, atunci rangul respectiv de adresă este considerat defect. Deşi este un test pentru identificarea liniilor de adresă blocate pe 1 sau pe 0 [SRIN78], în această formă poate detecta şi defecte de scurtcircuit între linii de adresă dacă este realizată funcţia SI cablat între ele.

Avantaje: utilizînd un cod corector de erori este evitată mascarea defectelor de adresă de către defecte ale liniilor de date sau ale C.I. de memorie cu condiţia ca numărul de erori dintr-un cuvînt să nu fie mai mare decît capacitatea de corecţie a codului.

Dezavantaje: este dificil de implementat pe testoarele de memorii deoarece necesită un codor şi un decodor combinaţional între generatorul de test şi memoria testată.

Acest test se pretează însă la diagnosticarea in vivo, de service a unităţilor de memorie, calculatoale în care acestea funcţionează putînd uşor realiza codarea şi decodarea. Viteza cu care se execută acest test nu este critică.

O analiză detaliată şi o generalizare a acestui algoritm este făcută în capitolul 4.

3.5.4. Defecte ale C.I. de memorie

După parcurgerea testelor de depanare de mai sus, logica de pe unitatea de memorie funcţionează corect. Urmează ca întreaga unitate să fie testată ca o memorie, funcţional, cu algoritmii de test prezentaţi în capitolul referitor la testarea C.I. de memorie.

In cazul memoriilor dinamice testele de depanare nu

verifică generarea semnalelor de regenerare. Datorită lungimii scurte şi a secvenţelor de adresare, aceste teste nu necesită regenerare. De aceea înainte de a trece la verificarea C.I. de memorie este necesară verificarea semnalelor de regenerare, de către operator, vizual cu osciloscopul sau cu sonda analizorului de semnături. Acest lucru este necesar pentru a face distincţie între C.I. de memorie defecte şi defecte ale logicii de regenerare.

Testele de performanţă mai pot scoate în evidenţă şi erori datorate nu C.I. de memorie ci unor circuite logice TTL prea lente, sau care introduc zgomote.

Alegerea algoritmilor de test, a tensiunilor şi relaţiilor temporale dintre semnale se face după o strategie care va fi discutată ulterior.

C.I. de memorie defecte sînt identificate pe baza BEM. Afişarea BEM se poate face fie după fiecare test fie cumulativ după un grup de teste sau după încheierea tuturor testelor.

Inlăturarea defectelor se face prin înlocuirea C.I. identificate ca fiind defecte.

3.6. Diagnoza asistată de calculator a unităţilor de memorie cu semiconductoare

Prezenta discuţie se referă la testoare specializate pentru memorii cu semiconductoare care sînt conduse de un mini sau microcalculator.

Un algoritm de diagnoză asistată de calculator a unităţilor de memorie a fost descris pentru prima oară de J.Chase în 1977 [CHAS77a], [CHAS77b]. Vom prezenta în continuare acest algoritm care este destinat testării pe fluxul de producţie.

Rezultatele testului de EXISTENCE (sub formă de BEM) şi a diagramei de adrese defecte (AFP) sînt interpretate de calculator. Calculatorul extrage proprietăţile figurilor formate de simbolurile de eroare din BEM şi le compară cu un număr de figuri standard ale căror proprietăţi sînt stocate în memoria sa.

Fie o memorie cu următoarele caracteristici:

- lungimea cuvîntului de date: 9 biţi
- lungimea cuvîntului adresă y, x : 12 biţi (4K)

C.I. de memorie cu linii separate pentru fiecare bit de adresă

- lungime cuvînt adresă z : 2 biţi (4 rînduri)
  (16 K cuvinte a 9 biţi)
- memoria are receptoare amplificatoare pentru datele de intrare şi registre (cu ieşire tri-state) pentru datele de ieşire; tactul acestor registre este generat din exteriorul plăcii
- există amplificatoare de adrese separate pentru fiecare rînd.

Lista figurilor de erori din BEM şi a defectelor asociate lor este dată în tabelul 3.1.

Tabelul 3.1.

Nr.	Figura de defecte	Defect probabil
1	Toate C.I. defecte	Defect general: -tensiuni de alimentare -circuit selecţie -lipsă tact selecţie, scriere sau date
2	Rînd 0,2	Defect pe linia A12
3	Rînd 1,3	Defect pe linia A13
4	Rînd 0	- decodor rînd i (C.I.2)
5	Rînd 1	- amplif. CS rînd i (C.I.1)
6	Rînd 2	- linia CS i blocată pe poziţia neselectat
7	Rînd 3	
8	Grupul col. 0-3	- amplif. DI 0-3 (C.I.6) - reg. DO 0-3 (C.I.7)
9	Grupul col. 4-7	- amplif.DI4-DI7 (C.I.8) - reg. DO 4-7 (C.I.9)
10	Col.8	- amplif.DI 8 (C.I.10) - reg.DO 8 (C.I.11) - defect pe una din liniile căii de date 8
11-18	Col.0-7	- defect pe una din liniile căii de date respective

După identificarea unei figuri de defecte calculatorul dă operatorului indicaţii asupra defectelor probabile şi căi-

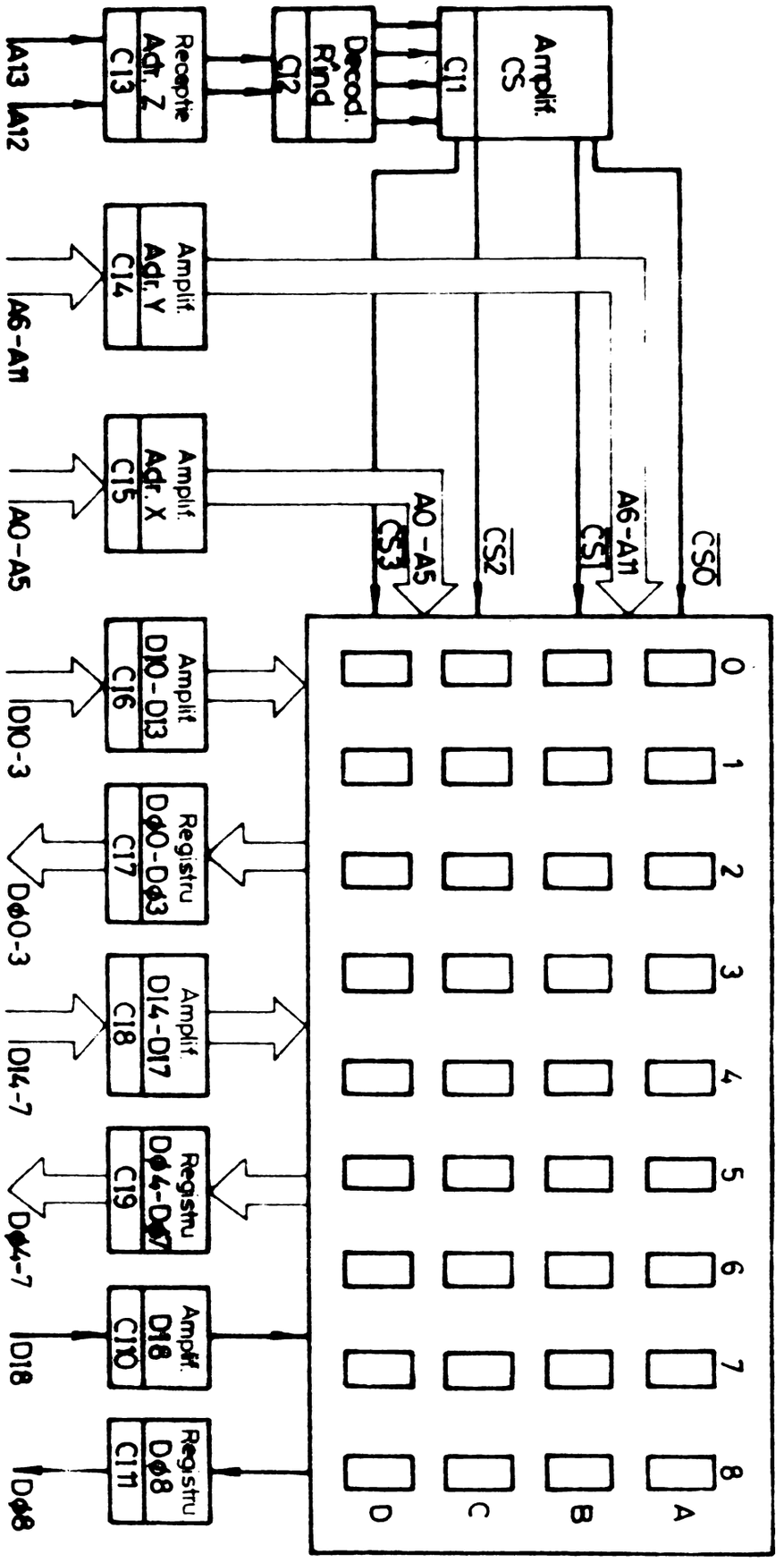


Fig. 3.9. Schema simplificată a unei memorii de 16K cuvinte de 9 biți realizată cu C.I. de 4K x 1

mulează ciclic unitatea testată în aşa fel încît semnalele să poată fi verificate. Verificarea se face fie vizual (cu osciloscopul) de către operator fie de către calculator cu o sondă care este conectată la un dispozitiv de evaluare a erorii. Acest dispozitiv poate fi construit pe următoarele principii

- principiul comparaţiei cu un semnal generat de testor;
- principiul numărării de tranziţii, de "1" sau zero;
- principiul analizei de semnături.

In ultimele două cazuri calculatorul indică operatorului punctele (nodurile) în care trebuie să aplice sonda, de la intrare spre ieşire. Semnătura (sau numărul de tranziţii) este comparată cu cea memorată anterior în acel punct prin palparea unei unităţi bune [MACR78]. Este probabil ca defectul să fie în elementul de circuit (activ sau pasiv) aflat între primul nod cu semnătură necorespunzătoare şi cel anterior. De asemenea, pentru fiecare nod sînt memorate cîteva semnături caracteristice anumitor defecte (blocat pe 0/1, scurtcircuit cu noduri fizic învecinate). Pe baza acestui dicţionar de semnături calculatorul poate identifica defectul însă cu o probabilitate diferită de 1.

Dezavantajul metodei de mai sus, (numită metoda sondei ghidate), este că necesită un efort foarte mare de programare.

[CHAS77a], [CHAS77b] recomandă o organigramă a unui program pentru testarea de producţie a memoriilor cu semiconductoare (fig.3.1o).

Testarea este începută prin execuţia testului March pentru a verifica funcţionarea memoriei. In cazul detectării unei erori se trece la execuţia testelor de depanare. Consider că nu este necesară trecerea la testele de depanare în cazul în care s-au detectat defecte într-un număr de C.I. de memorie mai mic decît numărul de rînduri. În acest caz este mai probabil ca să fie defecte C.I. de memorie respective.

Execuţia testelor de depanare

a. Se execută testul de existenţă.

Se trece la analiza formei figurii de simboluri de eroare, prin compararea proprietăţilor ei cu fiecare din fi-

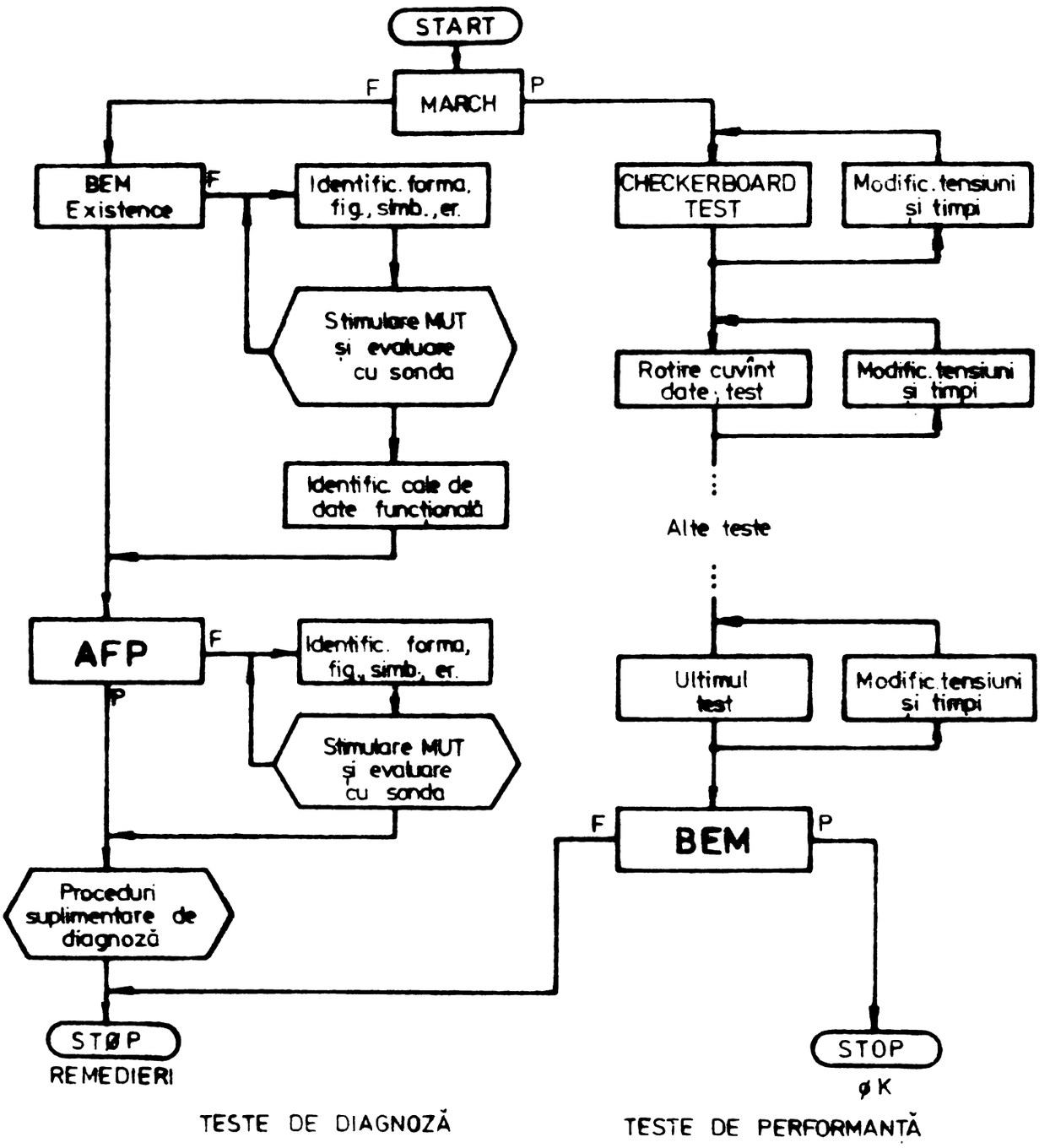


Fig. 3.10. Organigrama unui program pentru testarea cu depanare a unităților de memorie
[CHAS77a], [CHAS77b]

gurile stocate în memoria calculatorului. Pentru fiecare coincidenţă de proprietăţi se generează secvenţe de stimulare, se ghidează sonda şi se evaluează răspunsurile pînă la identificarea nodului defect.

b. Se caută o cale de date funcţională.

Căutarea începe cu calea de date cea mai îndepărtată de emiţătoarele de adresă.

c. Se execută testul de căi de adresă.

Se evaluează diagrama adreselor defecte (AFP), similar ca şi la evaluarea BEM. Se stimulează placa şi se evaluează răspunsurile pentru fiecare coincidenţă dintre proprietăţile formei figurii de simboluri de eroare cu cele memorate.

d. Se execută (eventual) teste suplimentare de diagnostic.

e. Se tipăresc instrucţiunile de remediere a plăcii şi se execută aceste remedieri.

Execuţia testelor pentru C.I. de memorie

Se execută întîi acele teste care au o durată proporţională cu N. Se înlocuiesc C.I. cu C.I. pretestate şi se trece la execuţia testelor cu durată mare. Aceste teste durează în general peste 15 min şi testorul poate funcţiona nesupravegheat.

Se înlocuiesc C.I. cu defecte şi se repetă testele cu durată mare.

3.7. Strategii de testare a UM cu semiconductoare

Aşa cum s-a arătat în capitolul 2 (defecte ale C.I. de memorie), o testare completă a memoriilor RAM nu este posibilă, prohibitiv fiind timpul necesar testării. O testare de producţie trebuie să fie în primul rînd economică (fiecare UM să ocupe cît mai puţin timp testoarele) şi eficientă (să detecteze cu maximum de probabilitate un număr cît mai mare de defecte, asigurînd astfel o fiabilitate acceptabilă în exploatare).

Prin strategii de testare se înţeleg toate acţiunile întreprinse în vederea stabilirii unei metodologii de testare

care să fie atît eficientă cît şi rentabilă din punct de vedere economic.

Paragrafele anterioare au fost consacrate metodologiilor de identificare rapidă a defectelor majore ale UM, în special cele ale interfeţei dintre calculator şi matricea de C.I. După asigurarea funcţionării corecte a acesteia rămîne problema identificării C.I. de memorie cu defecte. Se acceptă în literatură o clasificare a acestora în defecte ferme (care pot fi detectate de teste scurte) şi defecte cu latenţă mare, în această categorie intrînd în special defectele legate de sensibilitatea la informaţia înscrisă, sensibilităţi ale amplificatoarelor de citire, etc.

Avînd în vedere costul foarte ridicat al echipamentelor de testare, durata de execuţie a unor teste este un criteriu esenţial în alegerea unui program de test. Programe lungi duc pe de o parte la reducerea capacităţii de testare exprimată în unităţi/oră iar pe de altă parte la creşterea preţului de cost.

Din motivele de mai sus trebuie executaţi numai acei algoritmi de test care detectează defecte a căror probabilitate de manifestare este suficient de mare (cu latenţă mică). Aceşti algoritmi trebuie executaţi la acele valori ale tensiunilor de alimentare şi ale relaţiilor de timp între semnalele cu care este stimulată UM, pentru care probabilitatea de manifestare a defectelor este maximă (minimizarea latenţei). Identificarea acestor defecte probabile (specifice fiecărui lot de circuite integrate) şi a condiţiilor de stimulare în care se manifestă cu probabilitate maximă se numeşte caracterizarea C.I. de memorie.

Caracterizarea C.I. de memorie [FEI78]

Un studiu de caracterizare constă din cel puţin două faze:

a.- un studiu al interdependenţei între oricare doi parametri (timpi, tensiuni de alimentare, nivele de intrare şi ieşire);

b.- un studiu al sensibilităţii la informaţie.

Un studiu minimal al interdependenţei între parametri constă în analiza fiecărui parametru de timp în raport cu fie-

care tensiune de alimentare, de asemenea, studiul interdependenţei între oricare două tensiuni de alimentare. Pentru 14 parametri de timp şi 3 tensiuni de alimentare, rezultă că pentru fiecare C.I. din lotul eşantion şi pentru fiecare test sînt necesare 29 de diagrame SHMOO. Unii producători utilizează pînă la 100 de diagrame SHMOO pentru caracterizare [FEL78].

Din analiza statistică a acestor diagrame se pot alege valorile parametrilor la care trebuie să fie executat fiecare test.

Pentru studiul sensibilităţii la informaţie există două metode utilizate frecvent. Prima constă în testarea diagramelor SHMOO, avînd ca variabile V_{DD} şi V_{BB}, pentru toate testele care au capacitatea de a detecta astfel de defecte şi a urmări deosebirile dintre zonele de funcţionare fără eroare.

A doua metodă constă în căutarea valorilor minime şi maxime ale unui parametru, între care un test nu detectează erori.

Testele foarte lungi, ca de pildă GALPAT nu pot fi utilizate în testarea de producţie însă ele sînt strict necesare în faza de evaluare, pentru a verifica faptul că nu există defecte care să nu fie detectate de testele mai scurte însă pe care testele lungi le detectează.

De obicei se mai introduce un al treilea studiu în cazul în care se constată anomalii ale zonelor de funcţionare.

Defectele pe care diverşi algoritmi de test le descoperă cu o probabilitate mai mare sînt prezentate sintetic în tabelul A2.10 din anexa A2.

Este util ca această caracterizare a C.I. de memorie să fie făcută înainte de proiectarea UM deoarece se poate ţine seama de valorile maxime şi minime ale fiecărui parametru.

Uneori rezultatele acestor caracterizări sînt publicate sau comunicate [IBM82], [DANC83f] însă în general atît marile firme producătoare cît şi cele utilizatoare de C.I. de memorie păstrează secrete aceste caracterizări.

După cum s-a arătat anterior este improbabil să poată fi conceput un test practic implementabil şi care să detecteze toate defectele posibile. In plus, chiar dacă se face un studiu de caracterizare amănunţit, pe baza căruia se stabileşte un program de test care să descopere toate defectele probabi-

le, furnizorii schimbă frecvent măştile, tehnologiile sau programele lor de testare, astfel încît, după un interval de timp caracterizarea făcută nu mai este valabilă.

Este necesar de aceea ca, avînd un program de test pentru control intrare, să fie urmărite orice modificări ale procentului de C.I. de memorie respinse. Analiza pieselor respinse poate duce la îmbunătăţirea programelor de test. Caracterizarea trebuie repetată suficient de des pentru a putea sesiza diferenţe între laturile de C.I. In multe cazuri, diferenţe ale caracteristicilor C.I. de memorie , chiar dacă acestea se menţin în limita specificaţiilor furnizorului, crează probleme la nivel de sistem.

Caracterizarea trebuie de asemenea, repetată ori de cîte ori se schimbă furnizorul sau creşte numărul de unităţi de memorie returnate de beneficiari şi care nu sînt detectate ca defecte de programul de test. De asemenea, unele erori "soft" aleatoare pot fi cauzate de degradarea în timp a C.I. de memorie, sau pot apărea mai frecvent în anumite condiţii de funcţionare (tensiuni, timpi, informaţie şi secvenţă de adresare) în care unitatea de memorie nu a fost testată.

3.8. Concluzii

Utilizarea pe scară largă a C.I. de memorie în realizarea unităţilor de memorie se datoreşte evoluţiei rapide a tehnologiilor care au permis pe de o parte creşterea numărului de celule de memorare pe o pastilă de siliciu iar pe de altă parte scăderea spectaculoasă a preţului pe bit.

Testarea acestor C.I. şi a unităţilor de memorie realizate cu ele ridică probleme deosebite de dificile, în primul rînd pentru că nu este posibil accesul direct la celula de memorare ca în cazul feritelor.

Testarea lor exhaustivă nu este posibilă deoarece timpul necesar pentru aceasta este exagerat de lung. Se practică de aceea testarea numai pentru anumite defecte a priori cunoscute ca probabile şi cu teste specifice pentru fiecare din blocurile funcţionale din C.I. şi din unităţile de memorie realizate cu acestea. Dacă defectele blocurilor funcţionale

pot fi testate cu algoritmii practic implementabili, cunoscuţi astăzi, problema defectelor de sensibilitate la informaţia înscrisă rămîne deschisă.

In aceste condiţii, pentru a putea executa o testare care să asigure o rată a erorilor suficient de mică la utilizator este necesară caracterizarea C.I. de memorie, adică determinarea defectelor probabile şi a condiţiilor în care acestea se manifestă cu o probabilitate mai mare. Testele pentru fiecare clasă de defecte vor fi executate numai în aceste condiţii.

Pentru asigurarea unei rate a erorilor cît mai mică, producătorul are în faţă două alternative:

- testarea intensivă
  - prestarea 100% a C.I. de memorie
    - bucată cu bucată pe testoare de C.I.
  - testarea intensivă a UM
    - prin implementare pe UM de test şi testarea pe testoare de UM
- implementarea codurilor corectoare/detectoare de erori cu codor / decodor combinaţional

Fig.3.11.

Testarea bucată cu bucată pe testoare de C.I. de memorie este deosebit de costisitoare datorită preţului foarte ridicat al acestora precum şi a timpului necesar testării.

Testarea prin implementarea pe UM de test ridică problema în ce măsură aceasta este identică cu cele în care aceste C.I. vor fi implantate. De exemplu soclurile măresc capacitatea parazită a liniilor din matricea de C.I. făcînd ca forma semnalelor de stimulare să fie diferită de cea din UM în care C.I. de memorie se implantează fără socluri.

Testarea intensivă a UM pune problema manoperei de înlocuire a C.I., în cazul unei frecvenţe mai ridicate a căderilor.

Pentru a asigura la utilizator o rată a erorilor deosebit de mică singura cale rămîne implementarea codurilor core-

toare de erori. Această cale poate fi adaptată şi pentru aplicaţii uzuale, atunci cînd probabilitatea de apariţie a defectelor este mare. Chiar dacă pe o UM sînt implantate în acest caz mai multe C.I. de memorie, consumul specific poate fi mai mic prin acceptarea unei rate de erori mai mari pentru fiecare C.I. în parte. Această cale impune însă o proiectare deosebit de îngrijită a UM din următoarele motive:

- ETA trebuie să aibă acces la toţi biţii din matricea de C.I. de memorie, pentru a putea testa matricea trecînd peste blocul de corecţie;

- fiabilitatea finală a acestor UM poate fi mai mică decît a celor fără corectori dacă blocul de corecţie nu are o fiabilitate deosebit de ridicată [ELKI8o], [CLIFöo].

[ELKI8o] şi [FERR79] iau în considerare C.I. de memorie iniţial fără defecte şi calculează probabilitatea de funcţionare fără defecte. Nu există încă un model pentru cazul C.I. de memorie insuficient testate, însă consider că modelele citate se pot aplica, prin înlocuirea timpului de funcţionare fără defact a unui C.I. cu timpul de latenţă al defectelor.

[FERR79] face şi un studiu comparativ al numărului de C.I. de memorie înlocuite în cursul exploatării pentru variante cu şi fără corecţie de eroare. Se dă mai jos curba probabilităţii de funcţionare fără defecţiuni după [FERR79] (fig.3.12)

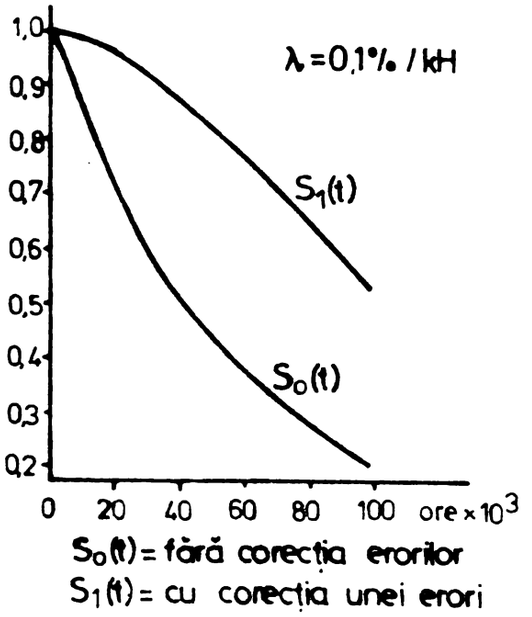
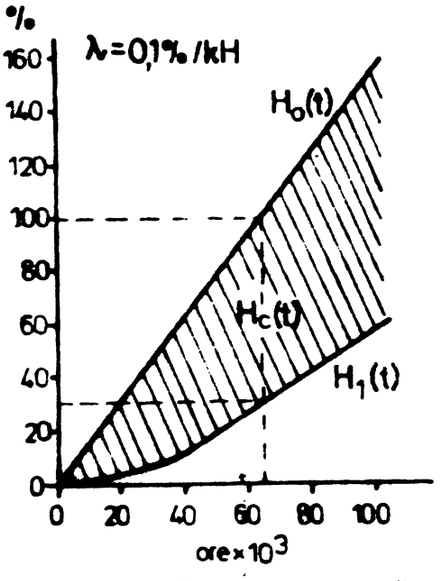


Fig.3.12. Siguranța în funcționare a UM



$H_0(t)$ - UM fără corecția erorii
$H_1(t)$ - UM cu corecția unei erori

Fig. 3.13. Procentul din populația inițială de CI care se înlocuiesc

In ceea ce privește testarea C.I. de memorie pe testoare specializate, achiziționarea unor astfel de sisteme de test (în valoare de peste 300.000 $) nu este întotdeauna justificată, nici pentru control intrare și nici pentru caracterizare. Firma Hewlet Packard raportează un procent de căderi la control intrare între 0,07% și 0,03% la C.I. de 16K în capsulă ceramică și între 0,12% și 0,33% pentru cele în capsulă de plastic [HP82]. In multe cazuri este mai economic un acord de testare cu firma producătoare sau cu o firmă specializată în testări, chiar dacă acest lucru duce la creșterea prețului unitar al C.I. Acest acord trebuie să prevadă neapărat comunicarea oricărei modificări în tehnologia de fabricație sau de testare.

In cazul în care producătorul de UM nu are astfel de acorduri cu producătorii de C.I. de memorii, introducerea în fabricație a unui nou lot de C.I. (mai ales la schimbarea furnizorilor) trebuie să fie precedată de caracterizarea lor. Această caracterizare presupune un volum de muncă foarte mare precum și un număr de minimum 200 ore-testor și ore-calculator. Timp în care se poate efectua o caracterizare nu este în general mai mic de o lună.

In cazul în care producătorul asigură și service pentru UM, trebuie arhivate <u>toate programele de test</u>, pe toată durata cît un UM testate cu acele programe se află în exploatare. O UM defectă, revenită pentru remediere va trebui testată cu programul cu care a fost testată în momentul fabricației, acela fiind adaptat detectării defectelor cu probabilitate maximă de apariție.

CAPITOLUL 4.

VERIFICAREA SI DIAGNOSTICAREA IN SISTEM A UNITATILOT DE MEMORIE RAM

4.1. Introducere

Testarea şi autodiagnosticarea sistemelor a stat întotdeauna în centrul atenţiei cercetătorilor din domeniul tehnicii de calcul. In ceea ce priveşte însă eficienţa testării în sistem a unităţilor de memorie RAM, consider că se poate obţine doar un coeficient de siguranţă relativ mic. Motivele care stau la baza acestei afirmaţii sînt expuse mai jos.

Sistemele de calcul nu dispun, în general, de surse separate şi programabile pentru alimentarea unităţilor de memorie, exersarea UM efectuîndu-se la o singură combinaţie a tensiunilor de alimentare, nu întotdeauna cea critică, necesară testării.

Relaţiile temporale între semnalele cu care este stimulată UM sînt întotdeauna aceleaşi şi nu cele critice. De exemplu, defecte care se manifestă prin dependenţa timpului de acces de adresa celulei nu pot fi detectate; în plus, temperatura de funcţionare nu este în general cea critică.

UM nu este exersată la viteza maximă la care ea funcţionează în sistem deoarece indiferent dacă acesta este o maşină von Neumann sau Wilkes, între ciclurile în care este adresată UM supusă testării se mai execută cicluri de scriere/citire din alte zone de memorie (cicluri de fetch, înscriere în registre rezultat etc.).

In afara cazului în care memoria dispune de registre de adresă, rularea testelor de tip GALLOPING, care execută toate salturile posibile de adresă, este nesemnificativă, deoarece între ciclurile în care este adresată UM testată sînt prezen-

te pe liniile de adresă şi alte combinaţii decît cele ce fac parte din secvenţa de test. Este adevărat că C.I. de memorie dinamice dispun, în general, de astfel de registre, însă aşa cum s-a arătat în anexa A2 testele de tip GALLOPING trebuie rulate fără regenerare pentru a nu se executa cicluri de memorie la alte adrese decît cele din secvenţa de test. Datorită frecvenţei reduse a ciclurilor la care este supusă UM testată, va fi cu siguranţă depăşit intervalul pentru care se garantează retenţia informaţiei.

Cu toate acestea testarea în sistem a UM poate detecta unele defecte majore apărute în intervalul de timp de la ultima testare pe un sistem specializat. Metoda este foarte importantă pentru service deoarece permite depanarea pe loc a UM, reducînd cu mult circulaţia plachetelor între beneficiari şi producători. Este însă important de menţionat că testarea şi diagnosticarea în sistem a UM nu poate substitui testarea pe echipamente specializate ; apariţia unor defecte de memorie în cursul operării sistemului, defecte care nu sînt detectate de programele de autotest, impune testarea UM pe testoarele specializate ale producătorului (sau ale întreprinderii de service).

Autorul a analizat programele de testare a memoriei pentru trei tipuri de calculatoare aflate în fabricaţie la ora actuală în R.S.R., fiecare reprezentînd o clasă

- Felix C-256 (512, 1024);
- Independent I-100 (PDP-11);
- MC-18.

Din această analiză a rezultat, în primul rînd că, testele incluse în aceste programe nu sînt cele mai eficiente. In programele de test a memoriei a clasei de calculatoare I-100 sînt incluse testele MARCH, GALPAT, GALLOPING WRITE RECOVERY, şi SHIFTING DIAGONAL. După cum s-a arătat mai sus, rularea "in vivo" a testului GALPAT, pe lîngă durata de execuţie excesiv de lungă, este nesemnificativă pentru detecţia acelor defecte pentru care a fost conceput, adică defecte de viteză a decodificatoarelor interne de adresă. Testele din clasa WRITE RECOVERY sînt şi ele nesemnificative datorită faptului că intervalul de timp dintre ciclurile de scriere şi ciclul de citire este mult mai mare decît timpul de ciclu

la care memoria este solicitată să lucreze în exploatare. Testul MARCH ar putea fi înlocuit cu teste din aceeaşi clasă mai recente şi mult mai performante (NAIR-A, NAIR-B sau MARCH-SUK)

Pe de altă parte, la familia calculatoarelor INDEPENDENT şi PDP-11 testele se desfăşoară pe pagini de 4K, ceea ce nu mai corespunde în condiţiile utilizării C.I. de 16Kx1. Pentru calculatorul I-102, a cărui memorie operativă este alcătuită din două module MMI-102, realizate cu C.I. de 16Kx1, aceste teste ar trebui modificate.

Din punctul de vedere al evaluării erorilor s-a constatat că, în mod practic, s-a rămas la metoda "oprit la eroare". Sînt afişate adresa erorii, cuvîntul aşteptat, cuvîntul citit şi secvenţa în care eroarea a fost detectată, [ITC**] interpretarea acestor date rămînînd în sarcina operatorului. Cu toate că evaluarea erorilor cu indicarea directă a C.I. în care au apărut (metoda hărţii erorilor pe modul BEM, prezentată în capitolul anterior) este deosebit de simplu de implementat software şi ar duce la sporirea vitezei de execuţie a testelor, ea nu este utilizată. Aceleaşi carenţe le prezintă şi programele de test a memoriei calculatoarelor Felix C-256 [CII72].

In testele de diagnoză a căilor de date şi adrese, sînt utilizate numai combinaţii de tip 2^j (00...010...0), care, aşa cum se va demonstra în acest capitol nu pot detecta defecte de scurtcircuit între liniile respective dacă defectele realizează funcţia logică SAU. Funcţia logică SAU cablat este realizată atunci cînd circuitele emiţătoare de pe modul sînt inversoare, ca în cazul modulului de memorie utilizat în calculatorul I-100.

Capitolul de faţă îşi propune să prezinte cîteva metode pentru diagnosticarea "in vivo" a unităţilor de memorie RAM cu semiconductoare, metode ce pot fi incluse în programele de autotest ale sistemelor de calcul.

4.2. Modelul de defecte

Modelul de defecte avut în vedere în acest capitol este, în linii mari, acelaşi cu cel prezentat în capitolul anterior.

Desigur pentru testarea "in vivo" se pot implementa metodele prezentate în capitolul 3. In acest capitol se va aborda o metodă prezentată de V.P.Srini, [SRIN78], în anul 1978.

[SRIN78] face următoarea clasificare a defectelor ce apar în UM de memorie RAM (fig.4.1).

Defecte
- -ale C.I. de memorie RAM
- -ale liniilor (cable faults)
  - -de date (Data cable)
  - -selecţie CS (chip selat cable CS
  - -adresă (address cable)
  - -control

Fig.4.1. Clasificarea defectelor în UM.

In cele ce urmează se va presupune că UM respectă schema bloc generală dată de [SRIN78], reprezentată în fig.4.2.

4.3. Căile de control şi date

4.3.1. Diagnosticarea liniilor de control

Aşa cum s-a arătat în capitolul anterior un defect pe liniile de control inhibă (în cazul general) funcţionarea memoriei în totalitate.

4.3.2. Diagnosticarea căilor de date

Una dintre metodele posibile ar fi implementarea testului EXISTENCE, [CHAS77a], [CHAS77b], prezentat în capitolul anterior.

[SRIN78] prezintă o metodă diferită dar care identifică cu precizie numai defecte de tip blocat pe 1 sau 0.

Experimentul S1

Pasul 1 - Se înscrie cuvîntul $W_0 = [00...0]$ într-o locaţie oarecare Y_j X_j dintr-un rînd de C.I.

Pasul 2 - Se înscrie cuvîntul $W_i = W_{i-1}+1$ în locaţia cu aceeaşi adresă x, y, din următorul rînd de C.I. şi se repetă pasul 2.

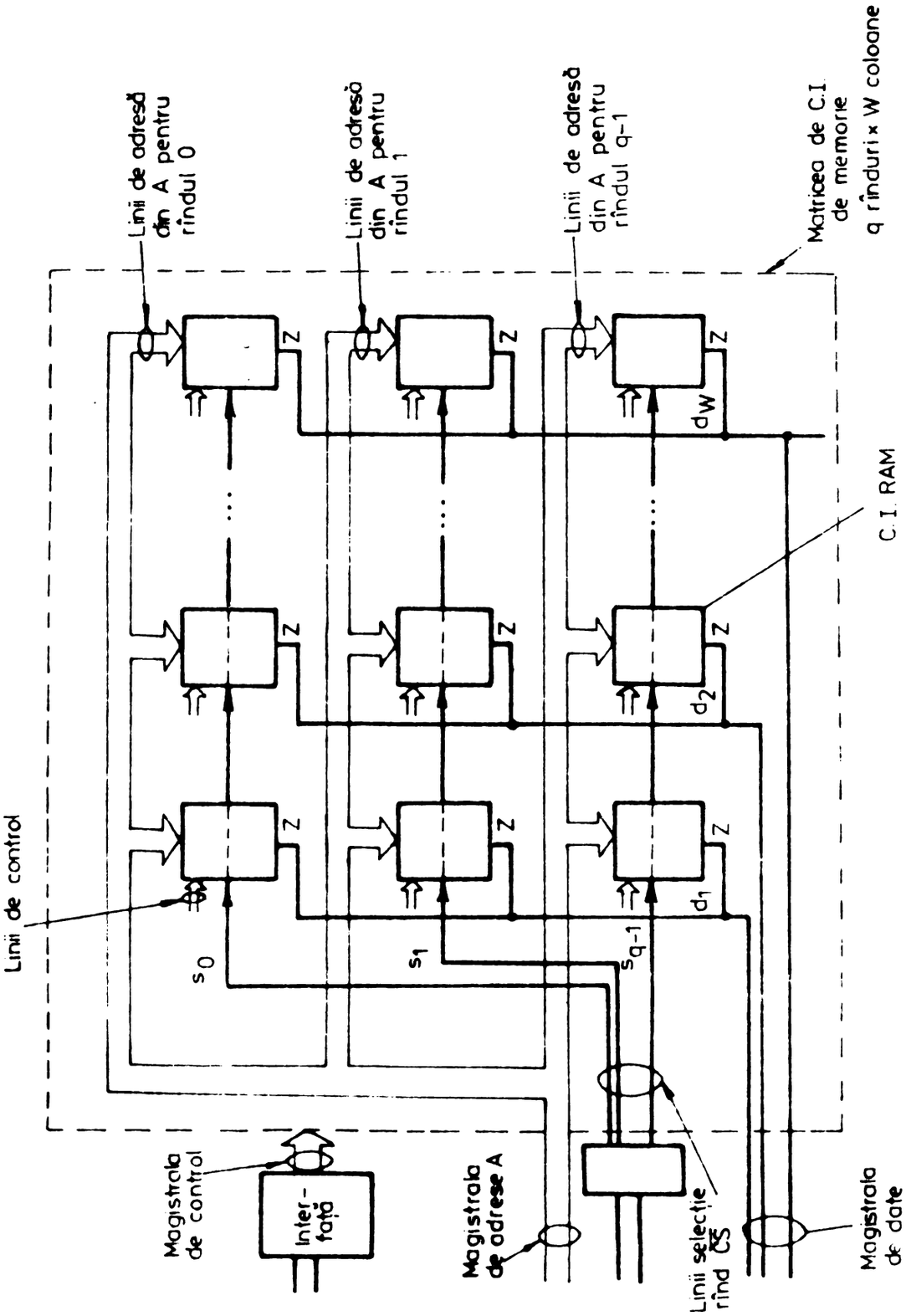


Fig.4.2.Unitate de memorie RAM [SRIN 78]

Pasul 3 - Se citesc toate locaţiile în ordinea în care au fost înscrise şi se formează o matrice MO din q rînduri şi W coloane, elementele matricii fiind cifre binare.

Pasul 4 - Se repetă paşii 1 - 3, însă cu date complementate.

Dacă linia de date i, $1 \leq i \leq W$ este blocată pe 0 (1), atunci coloanele i din matricile MO şi M1 vor fi formate numai din (0), (1).

Exemplul 4.1

Fie o UM cu q=8, W=8, 2p=1o. Matricile MO şi M1 sînt prezentate în tabelul 4.1.

TABELUL 4.1. Rezultatul experimentului S1 pentru următoarele defecte
- linia de date d2 blocată pe 1, detectat
- linia de date d4 blocată pe 1, detectat
- linia de date d8 blocată pe 0, detectat
- linia de date d7 în scurtcircuit cu lini d6, nedetectat

Adresa Z Y X	Cuvînt înscris 87654321	Cuvînt citit (MO) 87654321	Cuvînt înscris 87654321	Cuvînt citit (M1) 87654321
oooooooooooo	oooooooo	ooooloio	11111111	o1111111
ooloooooooooo	oooooool	oooololl	1111111o	o1111111o
oloooooooooo	ooooooio	ooooloio	1111111o1	o1111111
olloooooooooo	ooooooll	oooololl	11111100	o1111111o
looooooooooo	oooooloo	ooooll1o	11111o11	o1111o11
loloooooooooo	oooooio1	ooooll11	11111o1o	o1111o1o
lllooooooooo	ooooo11o	ooooll1o	11111oo1	o1111o11
lllloooooooo	ooooo111	ooooll11	11111ooo	o1111o1o
	defect: o ↔ 1 1		defect: o ↔ 1 1	

După cum se poate observa din acest exemplu, defectul de scurtcircuit între liniile d6 şi d7 nu este identificat deoarece în ambele cazuri informaţia citită coincide cu cea înscrisă. Cuvintele de date alese de [SRIN7o] sînt nesemnificative pentru defecte de scurtcircuit.

4.4. Diagnosticarea liniilor CS (Chip Select)

In cele ce urmează se va înţelege prin defect al unei linii $\overline{CS}$ blocarea ei pe 1. [SRIN7o]. In cazul blocării ei pe 0 rîndul respectiv de C.I. este selectat în permanenţă ceea ce duce la conflict pe magistrală în cazul unei citiri şi func-

ţionarea memoriei este inhibată.

Testul pentru liniile de CS propus de [SRIN78] se bazează pe faptul că interfaţa spre sistem a UM este uzual realizată cu C.I. TTL iar tentativa de citire dintr-un rînd de C.I. de memorie neselectat (a cărui linie de $\overline{CS}$ e blocată pe 1) duce întotdeauna la acelaşi rezultat (111...1 sau 000...0, funcţie de faptul dacă C.I. de interfaţă sînt neinversoare). In cazul de faţă vom presupune că citirea dintr-un rînd neselectat duce întotdeauna la rezultatul 111...1.

Experimentul propus de [SRIN78] pentru identificarea defectelor liniilor de $\overline{CS}$ se aseamănă cu experimentul S1, cu deosebirea că se utilizează un cod corector de erori, ceea ce permite executarea testului chiar şi în prezenţa unor defecte ale C.I. de memorie sau ale liniilor de date, cu condiţia ca numărul erorilor dintr-un cuvînt citit să nu fie mai mare decît capacitatea de corecţie a codului.

Experimentul S2.

Cuvîntul de informaţie va fi identic cu adresa pe axa z.

Pasul 1 - Se codifică cuvîntul de informaţie 0 şi se depune la o locaţie de memorie din rîndul de C.I. a cărui adresă pe axa z este 0.

Pasul 2 - Se incrementează cuvîntul de informaţie, se codifică şi se depune la locaţia de memorie cu aceeaşi adresă XY din rîndul următor de C.I.. Se repetă pasul 2 pentru toate rîndurile de C.I.

Pasul 3 - Se citeşte conţinutul fiecăreia din locaţiile de memorie în care s-au efectuat înscrieri, se decodifică, şi se formează matricea S.

Dacă linia $\overline{CS}_j$ este blocată pe 1 atunci în linia corespunzătoare din matrice se va găsi cuvîntul de informaţie corespunzător cuvîntului de cod 111...1 (sau a acelui cuvînt de cod aflat la cea mai mică distanţă Hamming de aceasta).

Exemplul 4.2

Fie o memorie cu q=4 şi W=8. Se utilizează un cod Reed Muller. Rezultatele sînt prezentate în tabelul 4.2 (după [SRIN78]).

TABELUL 4.2. Rezultatul experimentului S2 pentru defectul multiplu:
- linia CS0 blocată pe 1
- linia CS3 blocată pe 1
- linia D1 blocată pe 1
- linia D3 blocată pe 1
- defect de C.I. în rîndul 1 bit 7

Linia CS	Adresa	Biţi inf.	Cuvînt înscris 87654321	Cuvînt citit 87654321	Matricea S
$\overline{CS}_0$	oooooooooooo	oooo	oooooooo	11111111	looo *
$\overline{CS}_1$	oloooooooooo	oool	olololol	ooololol	oool
$\overline{CS}_2$	looooooooooo	oolo	oollooll	oollolll	oolo
$\overline{CS}_3$	llooooooooo	ooll	olloollo	11111111	looo *

Observaţii

In forma în care au fost prezentate de [SRIN78], testele pentru căile de date şi CS nu pot detecta defecte de scurtcircuit între liniile respective. Din motivul de mai sus, testul EXISTENCE în forma sa modificată, aşa cum a fost prezentat în cap.3 este superior din punctul de vedere al posibilităţilor de identificare a defectelor.

4.5. Diagnosticarea liniilor de adresă

Pentru diagnosticarea în sistem a UM poate fi utilizat testul propus de [CHAS77a], [CHAS77b], test care a fost prezentat în capitolul 3. In cele ce urmează va fi prezentată o generalizare a testului propus de [SRIN78] şi modificările efectuate de autor asupra acestuia. O analiză comparativă a celor două teste a fost prezentată de autor în [DANC83b], [DANC83d]. Tratarea formală a defectelor şi teoremelor din paragrafele următoare sînt originale.

[SRIN78] defineşte ca defect al liniilor de adresă blocarea pe 1 sau pe 0 a unora dintre aceste linii, cu excepţia a cel puţin unei linii.

Ca şi în testele S1 şi S2, [SRIN78] propune utilizarea unui cod corector de erori ceea ce permite desfăşurarea expe-

rimentului de test chiar şi în prezenţa unor defecte ale liniilor de date şi/sau defecte ale C.I. de memorie cu condiţia ca numărul erorilor dintr-un cuvînt citit să nu fie mai mare decît capacitatea de corecţie a codului utilizat.

Condiţia de mai sus se presupune a fi îndeplinită şi nu mai este enunţată în teoremele de mai jos. Demonstraţia teoremelor se va face în consecinţă pentru cazul în care informaţia este înscrisă în memorie necodificată.

4.5.1. Descrierea matematică a defectelor pe magistrale

Fie S mulţimea tuturor locaţiilor dintr-un rînd de C.I. al UM supuse testării.

A_j - adresa locaţiei aparent adresate. In cazul în care nu există defecte ale liniilor de adresă (sau C.I. RAM) A_j coincide cu adresa locaţiei efectiv adresate;

A_{SFj} - adresa locaţiei fizic adresate în prezenta defectelor de tip blocat pe 1 sau 0 dar în absenţa defectelor de tip scurtcircuit între liniile de adresă:

$$A_{SFj} = F_S(A_j) \qquad (4.1)$$

In cazul în care nu există defecte de tip blocat pe 1 sau 0 atunci

$$A_{SFj} = A_j$$

A_{BFj} - adresa locaţiei fizic adresate în prezenţa defectelor de blocare şi a defectelor de scurtcircuit:

$$A_{BFj} = F_B(A_{SFj}) \qquad (4.2)$$

In cazul în care nu există defecte de tip scurtcircuit

$$A_{BFj} = A_{SFj}$$

In cazul în care nu există nici defecte de blocare

$$A_{BFj} = A_j$$

Intotdeauna cînd natura defectelor este neesenţială se va utiliza pentru adresa fizică notaţia

$$A_{Fj} = F(A_j)$$

In cele ce urmează se va considera că C.I. de memorie prezintă o organizare de matrice pătrată cu un număr par de

liniii de adresă, 2p, acestea fiind numerotate de la 0 la $2p+1$.

In acest caz, se va considera

$$A_j = \sum_{i=o}^{2^{p-1}} a_{i,j}.2^i \qquad (4.3)$$

$$A_{SFj} = \sum_{i=o}^{2^{p-1}} \alpha_{i,j}.2^i \qquad (4.4)$$

$$A_{BFj} = \sum_{i=o}^{2^{p-1}} \beta_{i,j}.2^i \qquad (4.5)$$

Definiţia 1. Prin operaţia & (şi) între două adrese se înţelege o operaţie care realizează funcţia logică SI între cifrele binare de acelaşi rang ale operanzilor.

$$A_j \& A_k = \sum_{i=o}^{2^{p-1}} (a_{i,j}.SI.a_{i,k}).2^i \qquad (4.6)$$

Definiţia 2. Prin operaţia V între două adrese se înţelege o operaţie care realizează funcţia logică SAU între cifrele binare de acelaşi rang ale operanzilor

$$A_j \vee A_k = \sum_{i=o}^{2^{p-1}} (a_{i,j}.SAU.a_{i,k}).2^i \qquad (4.7)$$

Definiţia 3. Se va numi descriptor de blocare pe 0 numărul $L_{SF} \in S$, definit prin relaţia:

$$L_{SF} = \sum_{i=o}^{2^{p-1}} \ell_i.2^i$$

$$\ell_i = \begin{cases} 0 \text{ dacă linia i e blocată pe 0} \\ 1 \text{ dacă linia i nu e blocată pe 0} \end{cases} \qquad (4.8)$$

Definiţia 4. Se va numi descriptor de blocare pe 1 numărul $H_{SF} \in S$, definit prin relaţia:

$$H_{SF} = \sum_{i=o}^{2^{p-1}} h_i . 2^i$$

$$h_i = \begin{cases} 1 \text{ dacă linia i e blocată pe 1} \\ 0 \text{ dacă linia i nu e blocată pe 1} \end{cases} \qquad (4.9)$$

Se vor nota cu L_I şi H_I mulţimile indicilor liniilor de adresă blocate respectiv pe 0 sau 1.

$$L_I = \{ i \mid \ell_i = o \} \qquad (4.1o)$$

$$H_I = \{ i \mid h_i = 1 \} \qquad (4.11)$$

Deoarece o linie nu poate fi blocată atît pe 1 cît şi pe 0, rezultă că

$$L_I \cap H_I = \emptyset \qquad (4.12)$$

Definiţia 5. Adresa fizică a locaţiei adresate în prezenţa defectelor de blocare este dată de relaţia

$$A_{SFj} = F_S(A_j) = A_j \,\&\, L_{SF} \vee H_{SF} \qquad (4.13)$$

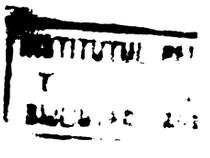
Definiţia 6. Prin defect de scurtcircuit între două linii de adresă u şi v se înţelege un defect în urma căruia cifrele binare corespunzătoare din adresa fizică sînt identice oricare ar fi valorile cifrelor binare respective din adresa aparentă.

$$\beta_u = \beta_v \qquad \forall a_u, a_v \in \{o,1\}$$

Proprietate: Relaţia de scurtcircuit este tranzitivă. Dacă linia u este în scurtcircuit cu linia v şi linia v este în scurtcircuit cu linia w atunci linia u este în scurtcircuit cu linia w. Proprietatea rezultă din tranzivitatea relaţiei de egalitate.

In cele ce urmează se va înţelege prin defect de scurtcircuit, o relaţie de scurtcircuit între mai multe linii de adresă, indiferent de numărul liniilor aflate în scurtcircuit.

O linie scurtcircuitată cu o linie blocată se va considera blocată, deci nu va apărea printre liniile afectate de defecte de scurtcircuit. Pot exista mai multe defecte de scurtcircuit. Fiecărui defect de scurtcircuit i se asociază o mulţime de indici B_x, alcătuită din indicii liniilor aflate în relaţia de scurtcircuit.



$$B_x = \left\{ i,k \,\middle|\, \beta_i = \beta_k \quad \forall\, a_i, a_k \in \{0,1\} \right\} \tag{4.14}$$

Fiecare mulţime B_x conţine cel puţin două elemente.

Lemă : Mulţimile B_x sînt disjuncte.

Demonstraţie prin reducere la absurd:

Fie B_{x1} şi B_{x2} şi $\exists\, v \in B_{x1} \cap B_{x2}$

Deoarece fiecărui defect de scurtcircuit i se asociază altă mulţime B_x, $B_{x1} = B_{x2}$ $\Rightarrow$

$$\exists\, u \in B_{x1} - (B_{x1} \cap B_{x2})$$

$$\exists\, w \in B_{x2} - (B_{x1} \cap B_{x2})$$

Conform proprietăţii de tranzivitate,

$$\left.\begin{array}{l} u \in B_{x1} \text{ şi } v \in B_{x1} \Rightarrow u.SC.v \\ v \in B_{x2} \text{ şi } w \in B_{x2} \Rightarrow v.SC.w \end{array}\right\} \Rightarrow$$

$$u.SC.w \Rightarrow w \in B_{x1} \text{ şi } w \in B_{x2}$$

ceea ce este absurd deoarece cele două mulţimi au fost asociate unor defecte diferite.

QED

Deoarece conform celor enunţate mai sus o linie nu poate fi simultan afectată de un defect de blocare şi de un defect de scurtcircuit, rezultă că:

$$\left(\bigcup B_x\right) \cap (H_I \cup L_I) = \emptyset \tag{4.15}$$

Definiţia 7. Se va numi descriptor al unui defect de scurtcircuit între liniile de adresă numărul $BF_x \in S$, definit prin relaţia

$$BF_x = \sum_{i=o}^{2^{p-1}} b_i \cdot 2^i$$

$$b_i = \begin{cases} 0, & i \notin B_x \\ 1, & i \in B_x \end{cases} \tag{4.16}$$

Numărul BF are toate cifrele binare corespunzătoare liniilor aflate în scurtcircuit egale cu 1. Există un descriptor pentru fiecare defect de scurtcircuit.

Vor fi luate în considerare numai acele defecte care realizează funcţia logică ŞI (dominanţă de 0) şi funcţia logică SAU (dominanţă de 1).

Pentru cazul funcţiei logice SI adresa fizică este dată de relaţia:

$$A_{BFj} = F_B(A_{SFj}) = \sum_{i=o}^{2^{p-1}} \beta_{i,j} \cdot 2^i$$

$$\beta_{i,j} = \begin{cases} a_{i,j} \& l_i \vee h_i, & \text{dacă nu } \exists\ B_x,\ i \in B_x \\ \underset{k \in B_x}{\&} a_{k,j}, & \text{dacă } \exists\ B_x,\ i \in B_x \end{cases} \qquad (4.17)$$

Pentru cazul funcţiei logice SAU, adresa fizică este dată de relaţia

$$A_{BFj} = F_B(A_{SFj}) = \sum_{i=o}^{2^{p-1}} \beta_{i,j} \cdot 2^i$$

$$\beta_{i,j} = \begin{cases} a_{i,j} \& l_i \vee h_i, & \text{dacă nu } \exists\ B_x,\ i \in B_x \\ \underset{k \in B_x}{\vee} a_{k,j}, & \text{dacă } i \in B_x \end{cases} \qquad (4.18)$$

Utilizînd descriptorii defectelor de scurtcircuit relaţiile de mai sus, pot fi puse şi sub forma:

Funcţia logică SI:

$$A_{BFj} = A_j \& L_{SF} \vee H_{SF} \& BA$$

$$BA = \begin{cases} \text{NOT } 0, & \text{dacă } A_j \& BF_x = BF_x \quad \forall x \\ \underset{y}{\&} (\text{NOT}(BF_y)), & \text{dacă } A_j \& BF_y \neq BF_y \end{cases} \qquad (4.19)$$

In operaţia $\underset{y}{\&}$ intră numai acei descriptori de scurtcircuit care îndeplinesc condiţia $A_j \& BF_y = BF_y$, adică cel puţin una din liniile aflate în scurtcircuit este "1" în adresa aparentă.

Funcţia logică SAU:

$$A_{BFj} = A_j \& L_{SF} \vee H_{SF} \vee BA$$

$$BA = \begin{cases} 0, & \text{dacă } A_j \& BF_x = 0 \quad \forall x \\ \underset{y}{\vee} BF_y, & \text{dacă } A_j \& BF_y \neq 0 \end{cases} \qquad (4.20)$$

In operaţia $\underset{y}{\vee}$ intră numai acei descriptori de scurtcircuit care îndeplinesc condiţia $A_j \& BF_y = 0$, adică cel puţin una din liniile aflate în scurtcircuit este "1" în adresa aparentă.

4.5.2. Generalizarea testului Srini pentru diagnoza liniilor de adresă

In cele ce urmează va fi prezentată o generalizare a testului propus de V.P.Srini în [SRIN78], însă cu notaţiile introduse de autor în paragraful 4.5.1. Prezentarea şi teoremele sînt originale. De menţionat că [SRIN78] a descris testul numai pentru identificarea liniilor blocate pe 1 sau 0.

Fie o mulţime de 2_p+1 adrese diferite, $\{A_0, A_1, \ldots, A_{2p}\}$, astfel încît să se realizeze o corespondenţă biunivocă între mulţimea indicilor liniilor de adresă $0,1,\ldots,2p-1$ şi mulţimea $\{A_0, A_1, \ldots, A_{2p-1}\}$.

Fie o mulţime de 2_p+1 cuvinte de informaţie distincte, astfel încît să se poată stabili o corespondenţă biunivocă între mulţimea cuvintelor de informaţie $\{D_0, D_1, \ldots, D_{2p}\}$ şi mulţimea numerelor $0,1,\ldots,2p-1,2p$.

Experimentul S3 [SRIN78].

Pasul 1. Se codifică cuvîntul de informaţie D_j, unde j are valoarea iniţială 0 şi se depune la locaţia A_j.

Pasul 2. Se incrementează j şi se repetă pasul 1 cît timp $j < 2p$.

Pasul 3. Se codifică cuvîntul de informaţie D_{2p} şi se depune la locaţia A_{2p}.

Pasul 4. Se citesc locaţiile A_j, $j=0,1,\ldots,2p-1$, se decodifică informaţia şi se alcătuieşte o matrice M_r din 2p elemente, numerotate $0,1,\ldots,2p-1$.
Dacă linia j din matricea Mr are valoarea D_{2p} în loc de D_j atunci linia de adresă j este defectă.

[SRIN78] a avut în vedere numai defecte de blocare şi a utilizat pentru desfăşurarea testului o mulţime particulară de locaţii avînd adresele:

$$\begin{cases} A_j = 2^j & , \ 0 \leq j \leq 2p-1 \\ A_{2p} = 0 \end{cases}$$

şi cuvinte de informaţie de forma

$$D_j = (j+1) \bmod (2p+1)$$

[SRIN78] nu face o demonstraţie a capacităţii testului său de

detecţie a defectelor.

Utilizînd descrierea matematică a defectelor, introdusă în paragraful 4.5.1, se va demonstra matematic capacitatea testului Srini S3 de identificare a liniilor de adresă defecte prin blocare şi scurtcircuit. Această demonstraţie este valabilă pentru o clasă mult mai largă de teste, inclusiv testul Chase [CHAS77a], [CHAS77b]. Se va demonstra că, dacă se au în vedere numai defectele de blocare ca în [SRIN78], relaţia care defineşte mulţimea adreselor locaţiilor pe care se desfăşoară testul are o expresie mult mai generală.

4.5.3. Identificarea liniilor de adresă blocate pe 1 sau 0 cu algoritmul Srini generalizat

Teorema 1: (T1)

Dacă adresele locaţiilor pe care se desfăşoară experimentul S3 satisfac relaţia

$$A_{Fj} = A_{F2p} \quad \forall j \in L_I \cup H_I \cup (\cup B_x) \tag{4.21}$$

atunci experimentul S3 identifică toate liniile de adresă defecte.

Demonstraţie prin reducere la absurd:

Dacă linia j este defectă ($j \in L_I \cup H_I \cup (\cup B_x)$) şi ea nu este detectată, atunci conţinutul locaţiei A_{Fj} este diferit de conţinutul locaţiei A_{F2p}

$$\langle A_{Fj} \rangle \neq \langle A_{F2p} \rangle = D_{2p}$$

şi deoarece A_{F2p} a fost înscrisă ultima, rezultă că cele două adrese sînt în mod necesar diferite

$$A_{Fj} \neq A_{F2p}$$

QED

Teorema 2 (T2)

Cifrele binare ale adreselor aparente care îndeplinesc condiţia

$$A_{Fj} = A_{F2p} \quad \forall j \in L_I \cup H_I \cup (\cup B_x)$$

enunţată în teorema T1 satisfac relaţia:

$$a_{i,j} = \begin{cases} a_{i,2p} & \forall\, i \neq j,\ 0 \leq i \leq 2p-1,\ 0 \leq j \leq 2p-1 \\ \text{NOT } a_{i,2p}, & i = j \end{cases} \tag{4.22}$$

adică

$$A_j = A_{2p} \oplus 2^j \tag{4.23}$$

unde $\oplus$ semnifică operaţia SAU EXCLUSIV.

Demonstraţie.

Deoarece cele $2p+1$ adrese aparente sînt distincte, înseamnă că

$$\forall j,\ 0 \leqslant j \leqslant 2p-1\ ,\ A_j \neq A_{2p}$$

Cele două adrese A_j şi A_{2p} diferă prin cel puţin o cifră binară, deci

$$\exists j,\ 0 \leqslant j \leqslant 2p-1 \quad \exists i,\ a_{i,j} = \text{NOT } a_{i,2p}$$

Dacă linia j este defectă, ea poate fi singura linie defectă (prin blocare) şi acest lucru trebuie să ducă la egalitatea :

$$A_{Fj} = A_{F2p}$$

deci

$$\beta_{i,j} = \beta_{i,2p} \qquad \forall i,\ 0 \leqslant i \leqslant 2p-1$$

Dacă celelalte linii nu sînt defecte, atunci

$$\beta_{i,m} = a_{i,m} \qquad \forall i,m\ ,\ i \neq j,\ 0 \leqslant i \leqslant 2p-1$$
$$0 \leqslant m \leqslant 2p-1$$

şi deci

$$a_{i,j} = a_{i,2p} \qquad \forall i \neq j$$

Deoarece $\exists$ i, astfel încît $a_{i,j}$=NOT $a_{i,2p}$ rezultă că i=j, şi prin urmare

$$a_{i,j} = \begin{cases} a_{i,2p} & , i=j,\ 0 \leqslant i \leqslant 2p-1,\ 0 \leqslant j \leqslant 2p-1 \\ \text{NOT}(a_{i,2p}) & , i=j \end{cases}$$

QED

Reciproca nu este adevărată, decît pentru defecte de blocare.

Teorema 3

T3

Condiţia necesară şi suficientă pentru ca experimentul S3 să detecteze toate liniile de adresă blocate pe 1 sau 0 este

$$a_{i,j} = \begin{cases} \text{NOT } a_{i,2p} & , i=j,\ 0 \leqslant i \leqslant 2p-1 \\ a_{i,2p} & , i \neq j,\ 0 \leqslant i \leqslant 2p-1,\ 0 \leqslant j \leqslant 2p-1 \end{cases}$$

adică

$$A_j = A_{2p} \oplus 2^j,\ 0 \leqslant j \leqslant 2p-1$$

Demonstraţie.

a. Necesitatea

Fie linia j blocată pe 1 sau 0, adică $j \in L_I \cup H_I$.

Dacă linia i nu este defectă, atunci

$$\beta_{i,j} = a_{i,j} \quad \forall i$$

Dacă $a_{i,j} \neq a_{i,2p}$ rezultă că $\beta_{i,j} \neq \beta_{i,2p}$, deci

$$A_j \neq A_{2p}$$

QED

Condiţia este deci necesară.

b. Suficienţa

In continuarea demonstraţiei se va presupune întotdeauna că $j \in L_I \cup H_I$, deci linia este blocată şi nu se va mai specifica acest lucru.

Se vor analiza cele trei categorii de cifre binare din A_j
- blocate
- în scurtcircuit
- fără defecte

b1 : Cazul liniilor blocate $i \in L_I \cup H_I$.

Cifrele binare corespunzătoare liniilor blocate sînt identice în toate adresele fizice.

$$\beta_{i,j} = \beta_{i,2p} = \ell_i \vee h_i \quad \forall i \in L_I \cup H_I$$

b2 Cazul liniilor în scurtcircuit.

$$\forall i \in \bigcup B_x$$

$$\beta_{i,j} = f(a_{k_1,j}, a_{k_2,j}, \ldots, a_{k_{n_x},j})$$

unde $k_1, k_2, \ldots, k_{n_x} \in B_x$

$$\beta_{i,2p} = f(a_{k_1,2p}, a_{k_2,2p}, \ldots, a_{k_{n_x},2p})$$

Deoarece linia j nu poate fi în scurtcircuit cu altă linie ea fiind blocată,

$$j \notin \bigcup B_x$$

înseamnă că dacă

$$a_{i,j} = a_{i,2p} \quad \forall i \in \bigcup B_x \ (i \neq j)$$

atunci

$$\beta_{i,j} = \beta_{i,2p}.$$

Prin urmare şi cifrele binare corespunzătoare liniilor în scurtcircuit sînt identice atît în A_{Fj} cît şi în A_{F2p}.

b3 Cazul liniilor neafectate de defecte.

$$i \notin L_I \cup H_I \cup (\cup B_x)$$

In acest caz

$$\beta_{i,j} = a_{i,j} \quad \forall i \notin L_I \cup H_I \cup (\cup B_x)$$

şi

$$\beta_{i,2p} = a_{i,2p} \; \forall i \notin L_I \cup H_I \cup (\cup B_x)$$

Prin urmare dacă

$$a_{i,j} = a_{i,2p}$$

atunci

$$\beta_{i,j} = \beta_{i,2p} \quad \forall i \notin L_I \cup H_I \cup (\cup B_x).$$

In toate cele trei cazuri $\beta_{i,j} = \beta_{i,2p}$

$$\forall i,\ 0 \leq i \leq 2p-1,\ \forall j \in L_I \cup H_I$$

$$\beta_{i,j} = \beta_{i,2p}$$

şi deci

$$A_{Fj} = A_{F2p}$$

QED

[SRIN78] a propus desfăşurarea testului S3 pe o mulţime de locaţii ale căror adrese aparente sînt definite prin relaţia:

$$A_j = \begin{cases} 2^j, & 0 \leq j \leq 2p-1 \\ 0, & j = 2p \end{cases} \qquad (4.24)$$

Teorema 4 (T4)

Adresele aparente definite prin relaţia

$$A_j = \begin{cases} 2^j, & 0 \leq j \leq 2p-1 \\ 0, & j = 2p \end{cases}$$

satisfac condiţia enunţată în teorema T1, pentru orice linie blocată pe 1 sau 0

$$A_{SFj} = A_{SF2p} \quad \forall j \in L_I \cup H_I.$$

Demonstraţia 1.

Deoarece $A_{2p}=0$ şi $A_j=2^j = 0 \oplus 2^j$

rezultă că:

$$A_j = A_{2p} \oplus 2^j, \quad 0 \leq j \leq 2p-1$$

şi conform teoremei T3

$$\forall j \in L_I \cup H_I, \; A_{SFj} = A_{SF2p}$$

QED

Demonstraţia 2.

a. Cazul liniei blocate pe 0; $j \in L_I$

$$\forall j \in L_I,\ 2^j \,\&\, L_{SF} = 0$$

şi

$$A_{SFj} = 2^j \,\&\, L_{SF} \vee H_{SF} = 0 \vee H_{SF} = H_{SF}$$

Deci

$$\forall j \in L_I,\ A_{SFj} = H_{SF} \qquad (4.25)$$

b. Cazul liniei blocate pe 1; $j \in H_I$

$$\forall j \in H_I,\ 2^j \,\&\, L_{SF} = 2^j$$

şi

$$A_{SFj} = 2^j \,\&\, L_{SF} \vee H_{SF} = 2^j \vee H_{SF} = H_{SF} \qquad (4.26)$$

Comparînd cele două rezultate, obţinem

$$A_{SFj} = H_{SF} \quad \forall j \in L_I \cup H_I \qquad (4.27)$$

Adresa fizică ce corespunde adresei A_{2p}=o este tot H_{SF} deoarece

$$A_{SF2p} = 0 \,\&\, L_{SF} \vee H_{SF} = 0 \vee H_{SF} = H_{SF} \qquad (4.28)$$

Prin urmare, comparînd relaţiile (4.27) şi (4.28), rezultă

$$\forall j \in L_I \cup H_I,\ A_{SFj} = A_{SF2p} = H_{SF}$$

QED

Corolar C4

Experimentul S3 desfăşurat pe o mulţime de locaţii ale căror adrese aparente sînt definite prin relaţia

$$A_j = \begin{cases} 2^j, & o \leq j \leq 2p-1 \\ o, & j = 2p \end{cases}$$

identifică toate liniile de adresă blocate pe 1 sau 0.

Demonstraţie: Rezultă din teoremele T4 şi T1.

Exemplul 4.3.

Va fi redat în continuare exemplul 3 din [SRIN78].

Fie o UM RAM cu q=8 rînduri de C.I. de memorare de 4Kb (2p=12) şi lăţimea cuvîntului de date W=8 biţi. [SRIN78] utilizează un cod Hamming corector de o eroare şi detector de două erori.

Liniile de adresă a1, a3, a9 şi a11 de pe rîndul selec-

tat de $\overline{CSO}$ sînt blocate fie pe 1 fie pe 0 deoarece elementele corespunzătoare din matricea R_o sînt 0.

TABELUL 4.3 - EXPERIMENTUL S3

j	Adresa aparenta Ai	Adresa fizica F(Ai) 1 0 0 1	Cuv.cod inscris $ $$$	Cuv.citit $ $$$	Matrice R0 bin.	zec.
0	000 000000000001	000 100000000011	0110 1001	0111 1001	0001	1
1	000 000000000010	000 100000000010	1010 1010	0001 0000	0000	0
2	000 000000000100	000 100000000110	1100 0011	1101 0011	0011	3
3	000 000000001000	000 100000000010	1100 1100	0001 0000	0000	0
4	000 000000010000	000 100000010010	1010 0101	1011 0101	0101	5
5	000 000000100000	000 100000100010	0110 0110	0111 0110	0110	6
6	000 000001000000	000 100001000010	0000 1111	0001 1111	0111	7
7	000 000010000000	000 100010000010	1111 0000	1111 0000	1000	8
8	000 000100000000	000 100100000010	1001 1001	1001 1001	1001	9
9	000 001000000000	000 100000000010	0101 1010	0001 0000	0000	0
10	000 010000000000	000 110000000010	0011 0011	0011 0011	1011	11
11	000 100000000000	000 100000000010	0011 1100	0001 0000	0000	0
12	000 000000000000	000 100000000010	0000 0000	nu se executa citiri		

Nota - Cu $ au fost notati bitii de informatie ai codului Hamming.
Bitul de date 4 a fost considerat blocat pe 1.
Cu 1 si 0 au fost marcate liniile de adresa blocate pe 1 sau 0.

4.5.4. Identificarea liniilor de adresă în scurtcircuit cu testul Srini S3 generalizat

Deoarece defectele de tip scurtcircuit sînt cele mai frecvente, mai cu seamă în matricea de C.I. de memorie, unde liniile de adresă au trasee apropiate pe o distanţă relativ mare, autorul a studiat condiţiile în care testul S3 detectează şi scurtcircuitele între liniile de adresă, indicînd aceste linii ca defecte.

Conform teoremei T1, condiţia ca experimentul S3 să identifice toate liniile de adresă defecte este dată de relaţia (4.21), prezentată aici sub forma:

$$A_{Fj} = A_{F2p} \quad \forall j \in L_I \cup H_I \cup (\cup B_x) \qquad (4.21)$$

Se va demonstra în continuare că experimentul S3 poate detecta şi defectele de tip scurtcircuit.

Teorema 5 | **T5**

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SI, atunci adresele aparente definite prin relaţia

$$A_j = \begin{cases} 2^j & , \quad 0 \leq j \leq 2p-1 \\ 0 & , \quad j = 2p \end{cases}$$

satisfac condiţia

$$A_{BFj} = A_{BF2p} \quad \forall j \in B_x$$

(enunţată în teorema T1) pentru orice linie j aflată în relaţie de scurtcircuit cu un grup de linii.

Demonstraţie.

Fie linia j în scurtcircuit cu liniile $u_{v1}, u_{v2}, \ldots, u_{vw}$ şi deci $B_x = \{ j, u_{v1}, u_{v2}, \ldots, u_{vw} \}$

Deoarece conform relaţiei (4.24)

$$A_j = \begin{cases} 2^j & , \quad \forall j = 0,1,\ldots,2p-1 \\ 0 & , \quad j = 2p \end{cases}$$

rezultă că

$$a_{i,j} = 0 \quad \forall i \neq j$$

Deoarece j diferă de $u_{v1}, u_{v2}, \ldots, u_{vw}$, rezultă

$$a_{u_{v1},j} = a_{u_{v2},j} = \ldots = a_{u_{vw},j} = 0 \qquad (4.29)$$

Conform relaţiei (4.17) pentru calculul adresei fizice

$$\beta_{j,j} = \beta_{u_{v1},j} = \beta_{u_{v2},j} = \ldots = \beta_{u_{vw},j} =$$
$$= \&_{k \in B_x} a_{k,j} = 1 \,\&\, 0 \,\&\, 0 \ldots \&\, 0 = 0$$

Deci dacă funcţia logică realizată de defectele de scurtcircuit este SI, şi dacă linia j este afectată de un defect de scurtcircuit, atunci $\beta_{j,j}=0$.

Dacă $j \in B_x$ şi $A_j=2^j$ pot fi pe 1 logic numai liniile blocate pe "1" ($i \in H_I$) şi liniile afectate de defecte de scurtcircuit.

S-a arătat mai sus că liniile scurtcircuitate cu linia j, şi linia j sînt pe "0" logic. Alte grupuri de linii aflate în scurtcircuit pot fi numai pe 0 logic deoarece toate liniile din grupurile respective au valoarea 0 în adresa aparentă, mulţimile B_x fiind disjuncte.

Rezultă că sînt pe 1 numai liniile care sînt blocate pe 1, şi deci:

$$A_{BFj} = H_I$$

In cazul în care se adresează locaţia cu adresa A_{2p}, toate liniile din adresa aparentă sînt pe 0, deci şi toate grupurile de linii în scurtcircuit vor fi pe 0. Vor fi pe 1 numai liniile blocate pe 1, deci:

$$A_{BF2p} = H_i$$

Rezultă:

$$A_{BFj} = A_{BF2p}$$

QED

Corolar C5

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SI, atunci experimentul S3 desfăşurat pe o mulţime de locaţii cu adresele aparente definite prin relaţia

$$A_j = \begin{cases} 2^j, & 0 \leq j \leq 2p-1 \\ 0, & j = 2p \end{cases}$$

identifică toate liniile de adresă defecte.

Demonstraţie.

Conform corolarului C4 experimentul S3 identifică toate liniile blocate.

Conform teoremei T5, adresele A_j îndeplinesc condiţia teoremei T1 deci experimentul S3 identifică şi liniile aflate în scurtcircuit cu alte linii.

Deci experimentul S3 identifică toate liniile de adresă defecte.

QED

Teorema 6, de unicitate T6

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SI condiţia enunţată în teorema T1

$$A_{Fj} = A_{F2p} \quad \forall j \in L_I \cup H_I \cup (\cup B_x)$$

este îndeplinită numai de mulţimea adreselor aparente definite prin relaţia

$$A_j = \begin{cases} 2^j, & 0 \leq j \leq 2p-1 \\ 0, & j = 2p \end{cases}$$

Demonstraţie.

Datorită teoremei T2 este suficient să demonstrăm că $A_{2p} = 0$.

In cazul în care există defecte de scurtcircuit, cifrele binare ale adresei fizice A_{F2p}, corespunzătoare liniilor aflate în scurtcircuit se calculează cu relaţia:

$$\forall B_x, \text{ şi } \forall k \in B_x,$$

$$\beta_{k,2p} = a_{k,2p} \;\&\; \Big(\underset{\substack{j \in B_x \\ j \neq k}}{\&} a_{j,2p}\Big) \qquad (4.30)$$

Cifrele binare ale adreselor fizice A_{Fk}, se calculează cu relaţia:

$$\forall B_x, \quad \forall k \in B_x,$$

$$\beta_{k,k} = a_{k,k} \;\&\; \Big(\underset{\substack{j \in B_x \\ j \neq k}}{\&} a_{j,k}\Big)$$

Conform teoremei T2

$$a_{k,k} = \text{NOT}\,(a_{k,2p})$$

$$a_{\substack{j,k \\ j \neq k}} = a_{\substack{j,2p \\ j \neq k}}$$

Deci

$$\forall B_x, \quad \forall k \in B_x,$$

$$\beta_{k,k} = \text{NOT}(a_{k,2p}) \;\&\; \Big(\underset{\substack{j \in B_x \\ j \neq k}}{\&} a_{j,2p}\Big)$$

Deoarece $A_{Fk} = A_{F2p}$ rezultă că şi cifrele binare de acelaşi rang sînt identice, deci

$$\beta_{k,k} = \beta_{k,2p}$$

şi se pot egala cele două relaţii

$$\forall B_x, \quad \forall k \in B_x,$$

$$\text{NOT}(a_{k,2p}) \,\&\, \Big(\underset{\substack{j \in B_x \\ j \neq k}}{\&} a_{k,2p}\Big) = a_{k,2p} \,\&\, \Big(\underset{\substack{j \in B_x \\ j \neq k}}{\&} a_{k,2p}\Big) \qquad (4.31)$$

Relaţia de mai sus poate fi adevărată numai dacă

$$\forall B_x, \quad \forall k \in B_x, \quad \underset{\substack{j \in B_x \\ j \neq k}}{\&} a_{k,2p} = 0 \qquad (4.32)$$

Deci numai dacă

$$\forall B_x, \quad \forall k \in B_x, \quad \exists j \in B_x, \; a_{j,2p} = 0$$

Deoarece relaţia de mai sus este adevărată pentru orice defect de scurtcircuit, rezultă

$$a_{k,2p} = 0 \qquad \forall k, 0 \leq k \leq 2p-1$$

Deci

$$A_{2p} = 0$$

QED

Cazul în care defectele de scurtcircuit realizează funcţia logică SAU

Pentru cazul funcţiei logice SAU, autorul a propus [DANC83b], [DANC83d], modificarea experimentului S3, prin desfăşurarea lui pe o mulţime de locaţii ale căror adrese satisfac relaţia

$$A_j = \begin{cases} \text{NOT } 2^j & , \ 0 \leq j \leq 2p-1 \\ \text{NOT } 0 & , \ j = 2p \end{cases} \qquad (4.33)$$

Se va demonstra mai întîi capacitatea testului de a identifica toate liniile blocate pe 1 sau 0. Următoarea teoremă este similară teoremei T5.

Relaţia (4.33) poate fi scrisă şi sub forma

$$\begin{cases} A_j = \sum_{i=0}^{2p-1} a_{i,j} \cdot 2^i \\ a_{i,j} = \begin{cases} 1 & , \ \forall i \neq j \\ 0 & , \ i = j \end{cases} \end{cases} \qquad (4.34)$$

Teorema 7 (T7)

Adresele aparente definite prin relaţia

$$A_j = \begin{cases} \text{NOT } 2^j & , \ 0 \leq j \leq 2p-1 \\ \text{NOT } 0 & , \ j = 2p \end{cases}$$

satisfac condiţia enunţată în teorema T1 pentru orice linie blocată pe 1 sau 0,

$$A_{SFj} = A_{SF2p} \qquad \forall j \in L_I \cup H_I$$

Demonstraţia 1.

Funcţia NOT 2^j se poate exprima ca

$$[\text{NOT}(0)] \oplus 2^j = \text{NOT } 2^j \qquad (4.35)$$

şi deoarece

$$\begin{cases} A_j = \text{NOT } 2^j & 0 \leq j \leq 2p-1 \\ A_{2p} = \text{NOT } 0 \end{cases}$$

Rezultă că

$$A_j = (\text{NOT } A_{2p}) \oplus 2^j \ , \ 0 \leq j \leq 2p-1 \qquad (4.36)$$

Conform teoremei T3, acest lucru implică:

$$A_{SFj} = A_{SF2p} \qquad \forall j \in L_I \cup H_I$$

QED

Demonstraţia 2.

Dacă linia j este blocată pe 0, atunci

$$A_j \,\&\, L_{SF} = (\text{NOT } 2^j) \,\&\, L_{SF} = L_{SF}$$

deoarece cifra binară j este 0 atît în A_j cît şi în L_{SF}

$$A_{SFj} = A_j \,\&\, L_{SF} \vee H_{SF} = L_{SF}$$

Dacă linia j este blocată pe 1, atunci

$$\alpha_{j,j} = 0 \,\&\, \ell_j \vee h_j = 0 \vee 1 = 1$$

Celelalte linii vor avea valoarea

$$\underset{i \neq j}{\alpha_{i,j}} = 1 \,\&\, \ell_i \vee h_i = \ell_i \vee h_i = \ell_i$$

Insă dacă linia j e blocată pe 1, atunci

$$\ell_j = 1, \; h_j = 1$$

deci

$$\alpha_{j,j} = \ell_j \vee h_j$$

$$\Downarrow$$

$$\alpha_{j,i} = \ell_i \vee h_i = \ell_i \qquad \forall i, \; 0 \leq i \leq 2p-1$$

$$A_{SFj} = L_{SF} \vee H_{SF} = L_{SF}$$

Toate cifrele binare ale adresei A_{2p} sînt "1"

$$a_{i,2p} = 1 \qquad \forall i, \; 0 \leq i \leq 2p-1$$

Adresa fizică va fi:

$$A_{SF2p} = A_{2p} \,\&\, L_{SF} \vee H_{SF} = L_{SF} \vee H_{SF} = L_{SF}$$

Comparînd cu A_{SFj}, rezultă

$$A_{SFj} = A_{SF2p}$$

QED

Corolar C7

Experimentul S3 desfăşurat pe o mulţime de locaţii ale căror adrese aparente sînt definite prin relaţia

$$A_j = \begin{cases} \text{NOT } 2^j, & 0 \leq j \leq 2p-1 \\ \text{NOT } 0, & j = 2p \end{cases}$$

identifică toate liniile de adresă blocate pe 1 sau 0.

Demonstraţie.

Rezultă din T7 şi T1.

Următoarea teoremă este similară teoremei T5.

Teorema 8 (T8)

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SAU, atunci adresele aparente definite prin relaţia

$$A_j = \begin{cases} \text{NOT } 2^j, & 0 \leq j \leq 2p-1 \\ \text{NOT } 0, & j = 2p \end{cases}$$

satisfac condiţia

$$A_{BFj} = A_{BF2p} \quad \forall j \in B_x$$

(enunţată în teorema T1) pentru orice linie j aflată în relaţie de scurtcircuit cu un grup de linii

Demonstraţie.

Demonstraţia este similară cu cea a teoremei T5. Fie linia j în scurtcircuit cu liniile $u_{v1}, \ldots, u_{vw}$ şi

$$B_x = \{j, u_{v1}, u_{v2}, \ldots, u_{vw}\}$$

Deoarece conform relaţiei (4.34)

$$a_{i,j} = 1 \quad \forall j \neq i$$

rezultă că

$$a_{u_{v1},j} = a_{u_{v2},j} = \ldots = a_{u_{vw},j} = 1 \qquad (4.37)$$

Conform relaţiei (4.18) pentru calculul adresei fizice

$$\beta_{j,j} = \beta_{u_{v1},j} = \beta_{u_{v2},j} = \ldots = \beta_{u_{vw},j} = \bigvee_{k \in B_x} a_{k,j} = \beta_{j,j} \vee 1 = 1$$

Prin urmare semnalele de pe linia β_j şi de pe liniile cu care este în scurtcircuit vor fi 1.

Din motivele expuse la demonstraţia teoremei T 5, pot fi 0 numai acele linii care sînt blocate pe 0, deci, dacă linia j este în scurtcircuit cu alte linii, atunci

$$A_{BFj} = L_{SF}$$

In cazul adresei A_{BF2p}, toate liniile din adresa aparentă fiind 1, de asemenea

$$A_{BF2p} = L_{SF}$$

Rezultă

$$A_{BFj} = A_{BF2p}$$

QED

Corolar C8

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SAU, atunci experimentul S3 desfăşurat pe o mulţime de locaţii cu adresele aparente definite prin relaţia

$$A_j = \begin{cases} \text{NOT } 2^j, & o \leq j \leq 2p-1 \\ \text{NOT } 0, & j = 2p \end{cases}$$

identifică toate liniile de adresă defecte.

Demonstraţie.

Conform corolarului C7 experimentul S3 identifică toate liniile de adresă blocate.

Conform teoremei T8, adresele A_j îndeplinesc condiţia teoremei T1, deci experimentul S3 identifică şi liniile aflate în scurtcircuit cu alte linii.

Experimentul S3 identifică toate liniile de adresă defecte.

QED

Teorema 9, de unicitate T9

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SAU, condiţia enunţată în teorema T1

$$A_{Fj} = A_{F2p} \quad \forall j \in L_I \cup H_I \cup (\cup B_x)$$

este îndeplinită numai de mulţimea adreselor aparente definite prin relaţia:

$$A_j = \begin{cases} \text{NOT } (2^j), & o \leq j \leq 2p-1 \\ \text{NOT } 0, & j = 2p \end{cases}$$

Demonstraţie.

Datorită teoremei T2 este suficient să demonstrăm că

$$A_{2p} = \text{NOT } 0$$

adică

$$a_{j,2p} = 1 \quad \forall j, \quad o \leq j \leq 2p-1$$

In cazul în care există defecte de scurtcircuit, cifrele binare ale adresei fizice A_{F2p}, corespunzătoare liniilor aflate în scurtcircuit se calculează cu relaţia:

$$\forall B_x, \quad \forall k \in B_x$$

$$\beta_{k,2p} = a_{k,2p} \vee \left(\bigvee_{\substack{j \in B_x \\ j \neq k}} a_{j,2p} \right)$$

Cifrele binare ale adreselor fizice A_{Fk} se calculează cu relaţia

$$\forall B_x, \quad \forall k \in B_x$$

$$\beta_{k,k} = a_{k,k} \vee \left(\bigvee_{\substack{j \in B_x \\ j \neq k}} a_{j,k} \right)$$

Conform teoremei T2:

$$a_{k,k} = \text{NOT } (a_{k,2p})$$

$$a_{\substack{j,k \\ j \neq k}} = a_{j,2p}$$

Deci relaţia devine

$$\forall B_x, \quad \forall k \in B_x$$

$$\beta_{k,k} = \text{NOT } (a_{k,2p}) \vee \left(\bigvee_{\substack{j \in B_x \\ j \neq k}} a_{j,2p} \right)$$

Deoarece $A_{Fk} = A_{F2p}$ rezultă că şi cifrele binare de acelaşi rang sînt identice, deci

$$\beta_{k,k} = \beta_{k,2p}$$

şi se pot egala cele două relaţii:

$$\forall B_x, \quad \forall k \in B_x$$

$$\text{NOT}(a_{k,2p}) \vee \left(\bigvee_{\substack{j \in B_x \\ j \neq k}} a_{j,2p} \right) = a_{k,2p} \vee \left(\bigvee_{\substack{j \in B_x \\ j \neq k}} a_{j,2p} \right)$$

Relaţia de mai sus poate fi adevărată numai dacă:

$$\forall B_x, \quad \forall k \in B_x \ , \left(\bigvee_{\substack{j \in B_x \\ j \neq k}} a_{j,2p} \right) = 1$$

adică:

$$\forall B_x, \quad \forall k \in B_x \ , \ \exists j, \ a_{j,2p} = 1$$

Pentru că relaţia de mai sus este adevărată pentru orice mulţime de indici B_x, oricare din linii putînd fi în scurtcircuit, rezultă că:

$$a_{j,2p} = 1 \quad \forall j, \quad 0 \leq j \leq 2p-1$$

QED

Exemplul 4.4. [DANC83b]

Fie o UM cu q=1, W=8, 2p=12. Fie liniile a2, a4 şi a8 blocate şi liniile a5, a6 în scurtcircuit. Pentru demonstrarea capacităţii de corecţie a codului se va considera linia de date d4 blocată pe 1.

Tabelul 4.4 redă desfăşurarea testului pentru cazul în care defectul de scurtcircuit realizează funcţia logică SI cablat iar tabelul 4.5, pentru cazul funcţiei logice SAU cablat.

Tabelul 4.4. Algoritmul Srini pentru funcţia SI cablat.

i	Adresă aparentă A_i	Adresă fizică $F(A_i)$	Cuvînt înscris	Cuvînt citit	Matrice R
0	oooooooooool	oooloooooolol	ollolool	ollllool	1
1	ooooooooolo	ooolooooollo	lolololo	lolllolo	2
2	oooooooooloo	oooloooooloo	llooooll	oooloooo	o
3	ooooooooolooo	oooloooolloo	lloolloo	llolllloo	4
4	oooooooloooo	oooloooooloo	loloolol	oooloooo	o
5	ooooooloooooo	oooloooooloo	olloollo	oooloooo	o
6	oooooloooooo	oooloooooloo	oooollll	oooloooo	o
7	ooooloooooooo	ooolloooooloo	llllooooo	llllooooo	8
8	oooloooooooo	oooloooooloo	loollool	oooloooo	o
9	ooloooooooooo	oolloooooloo	ololloolo	ololloolo	lo
lo	oloooooooooo	ololooooooloo	oolloolll	oolloolll	ll
ll	looooooooooo	looloooooloo	ooll1lloo	ooll1lloo	12
12	oooooooooooo	oooloooooloo	oooooooo	oooloooo	o

Tabelul 4.5. Algoritmul Srini pentru funcţia SAU cablat.

i	Adresă aparentă A_i	Adresă fizică $F(A_i)$	Cuvînt înscris	Cuvînt citit	Matrice R
0	111111111110	111111101110	ollolool	ollllool	1
1	111111111101	111111101101	lolololo	lolllolo	2
2	111111111011	111111101111	llooooll	oooloooo	o
3	111111110111	111111100111	lloolloo	llolllloo	4
4	111111101111	111111101111	loloolol	oooloooo	o
5	111111011111	111111101111	olloollo	oooloooo	o
6	111110111111	111111101111	oooollll	oooloooo	o
7	111101111111	111101101111	llllooooo	llllooooo	8
8	111011111111	111111101111	loollool	oooloooo	o
9	110111111111	110111101111	ololloolo	ololloolo	lo
lo	101111111111	101111101111	oolloolll	oolloolll	ll
ll	011111111111	011111101111	ooll1lloo	ooll1lloo	12
12	111111111111	111111101111	oooooooo	oooloooo	o

Se observă că poziţiile corespunzătoare din matricile rezultat sînt 0 pentru liniile de adresă defecte, fără a da posibilitatea identificării liniilor în scurtcircuit, separat de cele blocate.

Pentru o UM la care nu se cunoaşte a priori funcţia logică realizată de defectele de tip scurtcircuit se execută ambele variante ale experimentului S3.

Observaţie.

Sistemul de teoreme T1 - T9 are un grad mai mare de generalitate decît algoritmul S3 modificat. El poate fi aplicat practic tuturor algoritmilor de diagnoză a căilor de adrese, de exemplu algoritmul Chase, prezentat în capitolul 3.

Pînă în prezent testele de tip Chase erau executate intuitiv pe mulţimi de locaţii a căror adrese conţineau un singur 1 sau un singur 0. Grupul teoremelor originale T1 - T9 demonstrează că acestea sînt singurele combinaţii de adrese care permit identificarea tuturor liniilor de adresă defecte.

4.6. Un algoritm pentru identificarea defectelor de scurtcircuit între liniile de adresă

4.6.1. Necesitate şi definiţie

Algoritmii de test pentru depanarea asistată de calculator prezentaţi pînă acum, inclusiv algoritmul S3 (paragraful 4.5) în varianta originală şi în cea modificată de autor, pot identifica doar liniile de adresă defecte, fără a indica natura defectului.

Din datele citate în literatură [PHIL8o], cît şi din statisticile efectuate de ITC Timişoara, defectele cu frecvenţa cea mai mare şi mai dificil de identificat sînt cele de scurtcircuit. La unităţile de memorie MOS, care sînt organizate ca o matrice de C.I. de memorie, scurtcircuitele între liniile de adresă sînt foarte frecvente deoarece traseele acestor linii sînt paralele şi apropiate pe o distanţă relativ mare. De aceea consider că un algoritm care identifică nu numai liniile defecte ci şi defectele de scurtcircuit este deosebit de util.

In cele ce urmează se va considera că identificarea defectelor de scurtcircuit constă în determinarea tuturor mulţi-

milor B_x asociate lor. Algoritmul original ce va fi prezentat în cele ce urmează poate distinge liniile de adresă blocate de cele în scurtcircuit şi poate indica grupele de linii aflate în scurtcircuit.

4.6.2. Algoritmul SC

In cele ce urmează se va considera că funcţia logică realizată de defectele de scurtcircuit este fie SI, fie SAU şi ea este a priori cunoscută.

Se alege o mulţime de $2p+1$ locaţii de memorie distincte astfel încît să poată fi stabilită o corespondenţă biunivocă între cele 2p linii de adresă şi primele 2p locaţii.

Adresele acestor locaţii vor îndeplini condiţiile Cd1-Cd4.

Se vor utiliza de asemenea $2p+1$ cuvinte de informaţie distincte $D_0 \ldots D_{2p}$

Cd1 : Dacă linia j este blocată pe 1 sau 0, atunci

$$\forall j \in H_I \cup L_I \quad , \quad A_{Fj} = A_{F2p} \tag{4.38}$$

Cd2 : Dacă linia j este în scurtcircuit cu un grup de linii atunci

$$\forall j,x \quad j \in B_x \quad , \quad A_{Fj} = A_{F2p} \tag{4.39}$$

Cd3 : Dacă linia j nu este în scurtcircuit cu linia k chiar dacă acestea sînt afectate de defecte de scurtcircuit, atunci adresele fizice ale locaţiilor asociate celor două linii sînt diferite

$$\left.\begin{array}{l} j \in B_x \\ k \notin B_x \end{array}\right\} \Rightarrow \quad A_{Fk} \neq A_{Fj} \tag{4.40}$$

Cd4: Adresele fizice ale locaţiilor corespunzătoare liniilor de adresă care sînt scurtcircuitate între ele sînt identice

$$\forall j,m \in B_x \quad \Rightarrow \quad A_{Fj} = A_{Fm} \tag{4.41}$$

Experimentul SC.

Pasul 0: k are valoarea iniţială 0.

Pasul 1: Se codifică cuvîntul de informaţie D_j, unde j are valoarea iniţială k şi se depune la locaţia A_j.

Pasul 2: Se incrementează j (în mulţimea numerelor modulo 2p) şi se repetă pasul 1 de 2p ori.

Pasul 3 Se codifică cuvîntul de informaţie D_{2p} şi se depune la locaţia A_{2p}.

Pasul 4 Se citesc locaţiile A_j, $j=0,1,\ldots,2p+1$, se decodifică şi se alcătuieşte o matrice coloană M_k din 2p elemente, numerotate $0,1,\ldots,2p+1$.

Pasul 5 Se incrementează k şi se repetă paşii 1 - 4 cît timp $k < 2p$.

Dacă elementul j este D_j în toate cele 2p matrici coloană, atunci linia a_j nu este defectă. Dacă elementul j este D_{2p} în toate cele 2p matrici coloană, atunci linia a_j este blocată pe 1 sau 0.

Dacă elementul j este D_k, $k \neq j$ în cel puţin o matrice, atunci linia a_j este în scurtcircuit cu linia a_k. Cu alte cuvinte, dacă elementele dintr-o linie a celor 2p matrici nu sînt identice, atunci linia conţine toate elementele D_k, $k \in B_x$. Se pot identifica astfel toate liniile cu care linia a_j este în scurtcircuit.

Teorema 10 — T10

Experimentul SC identifică toate liniile de adresă blocate pe 1 sau 0 dacă adresele locaţiilor satisfac condiţia Cd1.

Demonstraţie.

Paşii 1 - 4 ai experimentului SC alcătuiesc un experiment S3. Din relaţia Cd1 teorema T1 rezultă că experimentul SC identifică toate liniile de adresă defecte prin blocare pe 1 sau 0.

QED

Teorema 11 — T11

Dacă linia j de adresă este în scurtcircuit cu un grup de linii de adresă atunci linia j din matricea rezultat conţine toate cuvintele de informaţie D_k, $k \in B_x$, corespunzătoare liniilor cu care este în scurtcircuit, şi numai pe acestea.

Demonstraţie.

Din condiţia Cd4 rezultă că la citirea locaţiei A_j se va obţine cuvîntul de informaţie D_m, dacă locaţia A_m a fost ultima locaţie din grupul liniilor scurtcircuitate între ele în care s-a executat o înscriere.

Deoarece ordinea în care sînt înscrise locaţiile se schimbă la fiecare pas printr-o deplasare cu 1, rezultă că

dintr-un grup de N locaţii, fiecare locaţie va fi înscrisă ultima cel puţin o dată.

Rezultă, din cele de mai sus că în linia j din matricea rezultat apar toate cuvintele de informaţie corespunzătoare liniilor scurtcircuitate cu linia j, inclusiv D_j.

Din cele expuse mai sus şi din condiţia Cd2 rezultă că

$$\langle A_{Fj} \rangle \neq \langle A_{F2p} \rangle$$

deci D_{2p} nu poate să apară în linia j din matricea rezultat.

Din condiţia Cd3 rezultă că dacă o linie j nu este în scurtcircuit cu linia k atunci $\langle A_{BFj} \rangle \neq \langle A_{BFk} \rangle$ şi deci D_k nu poate să apară în linia j a matricii rezultat.

Din ultimele două afirmaţii rezultă că o linie j a matricii rezultat este alcătuită numai din cuvinte de informaţie ale căror indici corespund unei singure mulţimi B_x, deci unui singur defect de scurtcircuit.

QED

Corolar C11 Experimentul SC desfăşurat pe o mulţime de locaţii ale căror adrese satisfac condiţiile Cd1 -Cd4 poate identifica toate defectele de tip scurtcircuit.

Demonstraţie. Rezultă din T11.

4.6.3. Alegerea adreselor aparente

Teoreme 12 T12 Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SAU atunci adresele aparente definite prin relaţia

$$A_j = \begin{cases} 2^j, & 0 \leq j \leq 2p-1 \\ 0, & j = 2p \end{cases}$$

satisfac condiţiile Cd1, Cd2, Cd3 şi Cd4.

Demonstraţie.

Condiţia Cd1 este îndeplinită conform teoremei T4.

Condiţia Cd2, dacă $j,u \in B_x$, atunci cifrele binare ale adresei A_j, vor fi

$$a_{j,j} = 1 \quad \text{şi} \quad a_{u,j} = 0$$

Deoarece $a_{j,j} = 1$ înseamnă că

$$\beta_{u,j} = a_{j,j} \vee a_{u,j} \vee \ldots = 1$$

In cazul adresei aparente A_{2p}

$$a_{k,2p} = 0 \quad \forall k \in B_x$$

şi

$$\beta_{u,2p} = a_{u,2p} \vee a_{j,2p} \vee \ldots = 0 \vee 0 \vee \ldots \vee 0 = 0$$

Deci

$$\beta_{u,j} \neq \beta_{u,2p}$$

de unde rezultă

$$A_{BFj} \neq A_{BF2p}$$

QED

Condiţia Cd3: conform relaţiei (4.24) şi (4.18), dacă linia k nu este blocată pe 1 ($k \notin H_I$)

$$\beta_{i,j} = 1 \quad \forall i,j \in B_x$$

$$\beta_{i,k} = 0 \quad \forall i \in B_x, \; k \notin B_x$$

Deoarece A_{Fj} şi A_{Fk} diferă prin cifra binară de rang i, rezultă

$$A_{Fj} \neq A_{Fk}$$

QED

Condiţia Cd4: Fiind dată mulţimea de indici B_x, conform relaţiei de definiţie (4.24), adresa aparentă conţine un singur 1 pe poziţia j şi deci

$$\left.\begin{array}{l} a_{i,j} = 0 \\ \beta_{i,j} = h_i \end{array}\right\} \forall (i,j), \; i \notin B_x, \; j \in B_x$$

Datorită scurtcircuitului şi funcţiei logice SAU

$$\forall i,j \in B_x \Rightarrow \beta_{i,j} = 1$$

şi atunci

$$\forall j \in B_x \qquad \beta_{i,j} = \begin{cases} h_i & , \; i \notin B_x \\ 1 & , \; i \in B_x \end{cases}$$

Din relaţia de mai sus rezultă că

$$\forall j,k \in B_x \qquad A_{Fj} = A_{Fk}$$

QED

Corolar C12

Dacă defcctele de scurtcircuit realizează funcţia logică SAU şi experimentul SC se desfăşoară pe o mulţime de 2p+1 locaţii cu adresele date de relaţia

$$A_i = \begin{cases} 2^i & , \quad 0 \leq i \leq 2p-1 \\ 0 & , \quad i = 2p \end{cases}$$

atunci experimentul identifică toate liniile de

adresă blocate pe 0 sau 1 şi toate defectele de scurtcircuit între liniile de adresă.

Demonstraţie. Rezultă din teoremele T1o, T11 şi T12

Teorema 13

T13

Dacă defcctele de scurtcircuit între liniile de adresă realizează funcţia logică SI, adresele aparente definite prin relaţia

$$A_j = \begin{cases} \text{NOT } 2^j & , \quad o \leq j \leq 2\rho-1 \\ \text{NOT } 0 & , \quad j = 2p \end{cases}$$

satisfac condiţiile Cd1, Cd2, Cd3 şi Cd4.

Demonstraţie.

Demonstraţia este similară demonstraţiei teoremei T12.

Condiţia Cd1 este îndeplinită conform teoremei T7.

Condiţia Cd2: Dacă $j,u \in B_x$, atunci cifrele binare ale adresei A_j vor fi

$$a_{j,j} = 0 \quad \text{şi} \quad a_{u,j} = 1$$

Deoarece defectul de scurtcircuit realizează funcţia logică SI, rezultă

$$\beta_{u,j} = a_{j,j} \,\&\, a_{u,j} \,\&\, \ldots = 0 \,\&\, a_{u,j} \,\&\, \ldots = 0$$

Adresa corespunzătoare adresei aparente A_{2p} va avea cifrele binare respective

$$a_{k,2p} = 1 \quad \forall k \in B_x$$

şi

$$\beta_{u,2p} = a_{u,2p} \,\&\, a_{j,2p} \,\&\, \ldots = 1 \,\&\, 1 \,\&\, \ldots \,\&\, 1 = 1$$

Deci

$$\beta_{u,j} \neq \beta_{u,2p}$$

de unde rezultă:

$$A_{BFj} \neq A_{BF2p}$$

ED

Condiţia Cd3: Conform relaţiilor (4.17) şi (4.33), dacă linia k nu e blocată pe 0 ($k \notin L_I$)

$$\beta_{i,j} = 0 \qquad \forall i,j \in B_x$$

$$\beta_{i,k} = 1 \qquad \forall i \in B_x,\ k \notin B_x$$

Rezultă deci, că

$$A_{Fj} \neq A_{Fk}$$

ED

Condiţia Cd4: Fiind dată mulţimea de indici B_x, corespunzătoare unui defect de scurtcircuit, conform relaţiei de definiţie (4.33), adresa aparentă A_j conţine un singur 0 pe poziţia j:

$$a_{i,j} = 1 \qquad \forall (i,j),\ i \notin B_x,\ j \in B_x$$

deoarece $i \neq j$. Deci

$$\beta_{i,j} = h_i$$

Datorită scurtcircuitului şi funcţiei logice SI realizate,

$$\forall i,j \in B_x\ , \qquad \beta_{i,j} = 0$$

şi deci

$$\forall j \in B_x\ , \quad \beta_{i,j} = \begin{cases} h_i & ,\ i \notin B_x \\ 0 & ,\ i \in B_x \end{cases}$$

deci

$$\forall j,\ k \in B_x, \quad A_{Fj} = A_{Fk}$$

QED

Corolar C13

Dacă defectele de scurtcircuit realizează funcţia logică SI şi experimentul SC se desfăşoară pe o mulţime de 2p+1 locaţii cu adresele date de relaţia

$$A_i = \begin{cases} \mathrm{NOT}(2^i) & 0 \leq i \leq 2p-1 \\ \mathrm{NOT}(0) & i = 2p \end{cases}$$

atunci experimentul SC identifică toate liniile de adresă blocate pe 0 sau 1 şi toate defectele de scurtcircuit între liniile de adresă

Demonstraţie. Rezultă din teoremele T1o, T11 şi T13.

In legătură cu alegerea adreselor aparente pentru experimentul SC se mai pot demonstra următoarele teoreme de unicitate.

Teorema 14 de unicitate T14

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SAU atunci numai adresele aparente definite prin relaţia

T13. Adresele

$$A_j = \begin{cases} 2^j, & 0 \leq j \leq 2p-1 \\ 0, & j = 2p \end{cases}$$

satisfac condiţiile Cd1, Cd2, Cd3 şi Cd4.

Demonstraţie.

Adresele definite prin relaţia de mai sus satisfac condiţiile Cd1 - Cd4, aşa cum s-a demonstrat în teorema T12. Demonstraţia faptului că numai aceste adrese îndeplinesc condiţiile Cd1 - Cd4 se va face prin reducere la absurd.

Fie

$$A_{2p} \neq 0$$

adică

$$\exists\, i,\ 0 \leq i \leq 2p-1,\ a_{i,2p} = 1$$

Conform teoremei T2 rezultă că:

$$a_{i,j} = 1 \quad \forall j \neq i$$

$$a_{i,i} = 0$$

Verificarea îndeplinirii condiţiei Cd4

Fie $B_x = \{i,j\}$

Cifrele binare i,j din adresa A_{Fj} vor fi

$$\beta_{i,j} = \beta_{j,j} = a_{i,j} \vee a_{j,j} = 1 \vee a_{j,j} = 1$$

Cifrele binare i,j din adresa A_{Fi} vor fi

$$\beta_{i,i} = \beta_{j,i} = a_{i,i} \vee a_{j,i} = 0 \vee a_{j,i} = a_{j,i}$$

şi conform teoremei T2

$$\beta_{i,i} = \beta_{i,j} = a_{j,2p}$$

Condiţia Cd4, care implică $\beta_{i,j} = \beta_{i,i}$ este îndeplinită $\forall B_x$ numai şi numai dacă:

$$a_{j,2p} = 1 \quad \forall j,\ 0 \leq j \leq 2p-1$$

adică:

$$\begin{cases} A_{2p} = \text{NOT } (0) \\ A_j = \text{NOT } (2^j), \quad 0 \leq j \leq 2p-1 \end{cases}$$

Pentru aceste adrese, conform teoremei T6,

$$A_{Fj} = A_{F2p} \quad \forall B_x, \quad \forall j \in B_x$$

deci nu este îndeplinită condiţia Cd2.

QED

Teorema 15, de unicitate

T15

Dacă defectele de scurtcircuit între liniile de adresă realizează funcţia logică SI atunci numai adresele aparente definite prin relaţia

$$A_j = \begin{cases} \text{NOT } 2^j, & 0 \leq j \leq 2p-1 \\ \text{NOT } 0, & j = 2p \end{cases}$$

satisfac condiţiile Cd1, Cd2, Cd3 şi Cd4.

Demonstraţie.

Adresele definite prin relaţia de mai sus satisfac condiţiile Cd1 - Cd4, aşa cum s-a demonstrat în teorema T13.

Demonstraţia unicităţii se va face prin reducere la absurd.

Fie A_{2p} = NOT (0), adică:

$$\exists i, \quad 0 \leq i \leq 2p-1, \quad a_{i,2p} = 0$$

Conform teoremei T2 rezultă că,

$$\left.\begin{aligned} a_{i,j} &= 0 \\ a_{i,i} &= 1 \end{aligned}\right\} \quad \forall j \neq i$$

Verificarea îndeplinirii condiţiei Cd4

Fie $B_x = \{i,j\}$.

Cifrele binare i,j din adresa A_{Fj} vor fi:

$$\beta_{i,j} = \beta_{j,j} = a_{i,j} \;\&\; a_{j,j}=0 \;\&\; a_{j,j}=0$$

Cifrele binare i,j din adresa A_{Fi} vor fi

$$\beta_{i,i} = \beta_{j,i} = a_{i,i} \;\&\; a_{j,i}=1 \;\&\; a_{j,i}=a_{j,i}$$

şi conform teoremei T2:

$$\beta_{i,i} = \beta_{j,i} = a_{j,2p}$$

Condiţia Cd4 care implică $\beta_{i,j} = \beta_{i,i}$ este îndeplinită $\forall B_x$ numai şi numai dacă:

$$A_{j,2p} = 0 \qquad \forall j, \quad 0 \leq j \leq 2p-1$$

adică pentru cazul particular

$$A_{2p} = 0$$

$$A_j = 2^j \qquad 0 \leq j \leq 2p-1$$

Pentru aceste adrese, conform teoremei T5:

$$A_{Fj} = A_{F2p} \qquad \forall B_x, \quad \forall j \in B_x$$

ceea ce contravine condiţiei Cd2.

QED

Exemplul 4.5. [DANC83b]

Fie o UM identică cu cea din exemplul 4.4, afectată de următoarele defecte:

- Liniile de adrese 2, 4 şi 8, blocate pe 1 sau 0
- Liniile 5, 6 şi 7 scurtcircuitate între ele.

Cuvîntul de informaţie asociat fiecărei linii de adresă este

$$D_j = \begin{cases} j + 1 & , \quad 0 \leq j \leq 2^{p-1} \\ 0 & , \quad j = 2p \end{cases}$$

Matricea rezultat este prezentată în tabelul 4.6.

Tabelul 4.6. Matricea rezultat a experimentului SC.

i*M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M11	M12	Observatii
0*1	1	1	1	1	1	1	1	1	1	1	1	
1*2	2	2	2	2	2	2	2	2	2	2	2	
2*0	0	0	0	0	0	0	0	0	0	0	0	linie blocata
3*4	4	4	4	4	4	4	4	4	4	4	4	
4*0	0	0	0	0	0	0	0	0	0	0	0	linie blocata
5*8+	8+	8+	8+	8+	8+	6	7+	8+	8+	8+	8+	sc.cu a6 si a7
6*8+	8+	8+	8+	8+	8+	6+	7	8+	8+	8+	8+	sc.cu a5 si a7
7*8	8	8	8	8	8	6+	7+	8	8	8	8	sc.cu a5 si a6
8*0	0	0	0	0	0	0	0	0	0	0	0	linie blocata
9*10	10	10	10	10	10	10	10	10	10	10	10	
*10*11	11	11	11	11	11	11	11	11	11	11	11	
*11*12	12	12	12	12	12	12	12	12	12	12	12	

Notă "+" = detecţie defect de scurtcircuit.

Conţinutul liniilor 2, 4 şi 8 este format numai din "0" deci liniile 2, 4 şi 8 sînt blocate pe 1 sau 0.

Liniile 5, 6 şi 7 sînt alcătuite din circuitele de informaţie 6, 7 şi 8 (D_5, D_6 şi D_7) deci liniile 5, 6 şi 7 sînt scurtcircuitate între ele.

4.6.4. Verificarea algoritmului SC prin simularea defectelor

Pentru verificarea posibilităţilor algoritmului SC de identificare a defectelor de blocare pe 1 sau 0 şi scurtcircuit acestea au fost simulate pe calculator. A fost utilizat un microcalculator tip TRS80 iar programele au fost scrise în

limbaj BASIC. Ele sînt prezentate în anexa A5a şi A5b.

Descriere:

După rezervările de memorie se face întîi calculul adreselor aparente conform relaţiei (4.24) (anexa A5a) sau a relaţiei (4.33) (anexa A5b). Calculul adreselor fizice se face cu ajutorul subrutinei 20000. Au fost simulate următoarele defecte

- linia 0, blocată pe 0
- linia 7, blocată pe 1
- liniile 1 şi 2 în scurtcircuit
- liniile 3, 4 şi 5 în scurtcircuit

Pentru defectele de mai sus, descriptorii sînt

LF = 11111110

HF = 10000000

BF1= 00000110

BF2= 00111000

Adresa fizică este calculată conform relaţiilor (4.19) pentru funcţia logică **SI** respectiv (4.20) pentru funcţia logică **SAU**.

Cele două programe diferă între ele doar prin liniile de program în care se definesc adresele aparente şi prin linia prin care se implementează defectul de scurtcircuit (20210).

A fost simulată o memorie de 256 cuvinte avînd prin urmare 8 linii de adresă.

Programul tipăreşte întîi tabelul cu cele 9 adrese aparente şi fizice pe care se desfăşoară experimentul SC. Se poate observa că în ambele cazuri A0 = A7 = A8 pentru liniile blocate şi A1 = A2 = A8, A3 = A4 = A5 = A8 pentru liniile aflate în scurtcircuit.

In matricea rezultat liniile 0 şi 7 sînt formate numai din 0, şi deci liniile de adresă 0 şi 7 sînt blocate pe 1 sau 0.

Liniile 1 şi 2 sînt formate numai din cuvintele de informaţie D1 = 2 şi D2 = 3, deci liniile de adresă 1 şi 2 sînt în scurtcircuit.

Liniile 3, 4 şi 5 sînt formate numai din D3=4, D4=5 şi D5=6 deci liniile de adresă 3, 4 şi 5 sînt în scurtcircuit.

Linia 6 este formată numai din D6=7 şi deci linia de

Tabelul 4.7. Rezultatele simulării experimentului SC cu programul din anexa A5a, pentru cazul în care defectele de scurtcircuit realizează funcţia logică SAU

RANG	ADRESE APARENTE bin	ADRESE APARENTE dec	ADRESE FIZICE bin	ADRESE FIZICE dec
A 0	00000001	1	10000000	128
A 1	00000010	2	10000110	134
A 2	00000100	4	10000110	134
A 3	00001000	8	10111000	184
A 4	00010000	16	10111000	184
A 5	00100000	32	10111000	184
A 6	01000000	64	11000000	192
A 7	10000000	128	10000000	128
A 8	00000000	0	10000000	128

MATRICE REZULTAT

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

REZULTATELE EXPERIMENTULUI SC

A 0 = LINIE BLOCATA
A 1 = SCURTC.INTRE 1 2
A 2 = SCURTC.INTRE 1 2
A 3 = SCURTC.INTRE 3 4 5
A 4 = SCURTC.INTRE 3 4 5
A 5 = SCURTC.INTRE 3 4 5
A 6 =
A 7 = LINIE BLOCATA

Tabelul 4.8. Rezultatele simulării experimentului SC cu programul din anexa A5b pentru cazul în care defectele de scurtcircuit realizează funcţia logică SAU. SI

RANG	ADRESE APARENTE bin	ADRESE APARENTE dec	ADRESE FIZICE bin	ADRESE FIZICE dec
A 0	11111110	254	11111110	254
A 1	11111101	253	11111000	248
A 2	11111011	251	11111000	248
A 3	11110111	247	11000110	198
A 4	11101111	239	11000110	198
A 5	11011111	223	11000110	198
A 6	10111111	191	10111110	190
A 7	01111111	127	11111110	254
A 8	11111111	255	11111110	254

MATRICE REZULTAT

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

REZULTATELE EXPERIMENTULUI SC

A 0	=	LINIE BLOCATA
A 1	=	SCURTC.INTRE 1 2
A 2	=	SCURTC.INTRE 1 2
A 3	=	SCURTC.INTRE 3 4 5
A 4	=	SCURTC.INTRE 3 4 5
A 5	=	SCURTC.INTRE 3 4 5
A 6	=	
A 7	=	LINIE BLOCATA

adresă 6 nu este defectă.

Această interpretare este executată în program între liniile 1640 - 1660, tipărirea fiind făcută în zona de program dintre liniile 1910 - 2010.

4.6.5. Implementarea algoritmului SC pe echipamente de testare automată a memoriilor

După cum se va arăta în capitolul 5 în care se face şi o analiză a arhitecturii echipamentelor de testare automată, testoarele specializate pentru UM au, în general, schema bloc din figura 6.1. Caracteristic acestora este existenţa unui procesor rapid, programabil care execută testele (test pattern) generînd cuvintele înscrise şi adresele conform unui program introdus în memoria sa de către mini/microcalculatorul care conduce procesul de testare. Evaluarea erorilor este făcută tot de către procesorul rapid,

Dacă facilitatea de evaluare a erorii nu poate fi inhibată, astfel încît calculatorul să poată genera toate cuvintele înscrise şi să poată avea acces la toate cuvintele citite, pentru a putea alcătui matricea rezultat, aplicarea testului SC de diagnoză a liniilor de adresă nu este posibilă.

Afirmaţia de mai sus se bazează pe de o parte pe faptul că nu a fost întîlnit în literatură un procesor rapid de test care să poată executa şi codarea şi decodarea Hamming şi pe de altă parte pe faptul că procesorul rapid ar trebui să aibă o memorie de date de cel puţin 2px2p octeţi pentru formarea matricii rezultat.

Implementarea testului SC este însă perfect posibilă pe testoare universale de plachete logice, cu schema bloc din figura 5.2, la care calculatorul furnizează toate cuvintele stimuli şi toate cuvintele răspuns. In acest mod calculatorul generează atît cuvintele de date cît şi cuvintele adresă, putînd executa algoritmul SC propus în § 4.6.2.

4.6.6. Implementarea algoritmului SC pe minicalculatorul I-100

In anexa A.4c este prezentat listingul de consolă şi programul de diagnoză a liniilor de adresă din modulele de memorie ale calculatorului I-100. El este destinat să completeze setul de programe DZKMA de verificare a memoriei RAM. Programele DZKMA au prevăzute teste pentru diagnoza liniilor de adresă, însă acestea nu indică decît faptul că o anumită linie de adresă este defectă. Interpretarea rezultatelor este greoaie şi, în plus, dacă defectele de scurtcircuit realizează funcţia logică SAU, acestea nu pot fi detectate de testele liniilor de adresă.

Calculatorul I-100 este echipat cu 8 module de memorie de 16 K cuvinte, de tip M12855, M3255 sau MS3255, produse de Fabrica de memorii electronice şi componente pentru tehnica de calcul Timişoara. Ele sînt realizate cu C.I. RAM de 4 kbiţi organizate în două matrici. Matricile de C.I. corespunzătoare octetului superior şi inferior au amplificatoare de adresă separate.

Una din primele dificultăţi întîmpinate s-a datorat faptului că în această familie de calculatoare adresa logică este adresa de cuvînt. Pentru a obţine cuvîntul de adresă efectiv aplicat de pinii blocului de memorie, adresa logică trebuie deplasată la dreapta cu o poziţie. Testarea liniilor de adresă din matricea octetului inferior (LSB) sau superior (MSB) se face executînd testul pe adrese (logice) pare sau impare.

Dacă K=(I+1) mod 12, se face înscrierea cuvîntului I+1 codat Hamming la adresa A(I) şi a cuvîntului "0" la adresa A_{2p}=A0. Se citesc apoi locaţiile $A_0, A_1, \ldots, A_{11}$, se decodifică şi se completează prima coloană a matricii rezultat (prin depunerea rezultatelor decodificărilor în cuvinte succesive, începînd cu adresa RIJ.

Se tipăreşte matricea rezultat apoi se trece la analiza ei. Dacă conţinutul liniei j din matricea rezultat este 0 se tipăreşte "LINIE BLOCATA". Dacă conţinutul este diferit de 0 sau j+1 atunci se tipăreşte conţinutul locaţiei respective

decrementat cu 1. Acesta este numărul liniei de adresă cu care linia de adresă curentă este în scurtcircuit.

Atît codarea cît şi decodarea, după un cod corector de o eroare şi detector de două erori se face prin tabele. Aceste tabele (TCHAM şi TDHAM) au fost obţinute cu ajutorul unui alt program scris în limbajul BASIC.

Subrutina GETADR : Execută calculul adresei la care trebuie făcută operaţiunea de scriere sau citire, funcţie de conţinutul registrului R3 şi de conţinutul locaţiei PARIMP. R3 conţine indicele adresei ce trebuie calculată (o,1,...,11,12). Adresa calculată este returnată în R5.

Subrutina CODARE : Execută codarea conţinutului registrului R4 (incrementat cu 1) după un cod Hamming corector de o eroare şi detector de două erori. In plus, cuvîntul astfel codat este înscris la adresa indicată de R5.

Subrutina DECOD : Execută decodarea conţinutului locaţiei de memorie indicată de R5. Rezultatul este returnat în R5, octetul inferior, atît pentru adrese pare cît şi pentru adrese impare.

Subrutina WRIJ : Completează matricea rezultat, înscriind conţinutul registrului R5, în poziţia corespunzătoare din matricea rezultat.

Subrutina TIPAR: Execută tipărirea matricii rezultat, iar BCBDMG execută conversia unui număr binar în zecimal ASCII.

După analiza matricii rezultat şi a tipăririi rezultatelor experimentului SC se înscrie 1 în locaţia PARIMP şi se execută procesul pentru adrese impare, începînd de la eticheta MAI .

4.7. Limite de aplicabilitate ale algoritmilor de diagnoză a liniilor de adresă

Limitele de aplicabilitate ale algoritmilor Srini modificat şi SC, precum şi a algoritmilor de tip Chase se referă la identificarea liniilor de adresă în scurtcircuit.

Emiţătoarele pentru semnalele de adresă pentru liniile de adresă din matricea de memorie sînt realizate, în general, în tehnologie TTL. Scurtcircuitele între ieşirile acestor

emiţătoare realizează funcţia logică SI datorită faptului că în stare 0 circuitele TTL pot absorbi un curent mult mai mare decît pot debita în starea 1 (16 mA în starea "0" faţă de 4oo uA sau 8oo uA în starea "1"). Dacă emiţătoarele sînt inversoare, atunci funcţia logică realizată este SAU (în raport cu semnalele de la intrarea emiţătoarelor)

$$\overline{\overline{a}_j \cdot \overline{a}_k} = \overline{\overline{\overline{a}_j} + \overline{\overline{a}_k}} = a_j + a_k$$

Limitările pot apare atunci cînd sînt în scurtcircuit mai multe linii de adresă, şi cînd nu se mai poate garanta funcţia logică SI cablat între ieşiri. Numărul limită de linii în scurtcircuit depinde de curentul de ieşire în scurtcircuit I_{os} al circuitelor emiţătoare.

A fost determinată experimental, pe un testor universal de C.I. logice, tip ENERTEC-SCHLUMBERGER, tip T925, dependenţa tensiunii de ieşire în nodul creat prin defectul de scurtcircuit funcţie de numărul de linii aflate în scurtcircuit şi de tensiunea de alimentare (fig.4.3) pentru un C.I. de tip 74Ho4 Dintre ieşirile S1 - S6 ale inversoarelor, numai una - S1 - se află în stare "0", celelalte fiind forţate în "1" (cazul cel mai defavorabil). Tensiunile au fost măsurate pentru un curent absorbit de 1 uA (sarcină MOS).

Se poate observa că pentru mai mult de trei linii aflate în scurtcircuit nivelul de "zero" intră în zona de indecizie. Astfel de defecte multiple (scurtcircuite între mai mult de trei linii) sînt foarte puţin probabile.

Programul cu ajutorul căruia au fost obţinute datele de măsurare este prezentat în anexa A5d şi este scris în limbaj BASCH.

O altă limitare este cauzată de rezistenţele ce sînt introduse în serie la ieşirea circuitelor emiţătoare, pentru atenuarea reflexiilor pe linii (fig.4.4). Ele au valori cuprinse între 1o şi 4o ohm.

Căderea de tensiune pe aceste rezistenţe produce alterarea nivelului logic pe liniile din matrice aflate în scurtcircuit pînă la valoarea de indecizie. In consecinţă vor fi adresate alte locaţii decît pe care ar trebui să se desfăşoare testele, adică A_{Fo} ... A_{F2p}, nu întotdeauna aceeaşi la două operaţii pe aceeaşi adresă aparentă. Pentru a

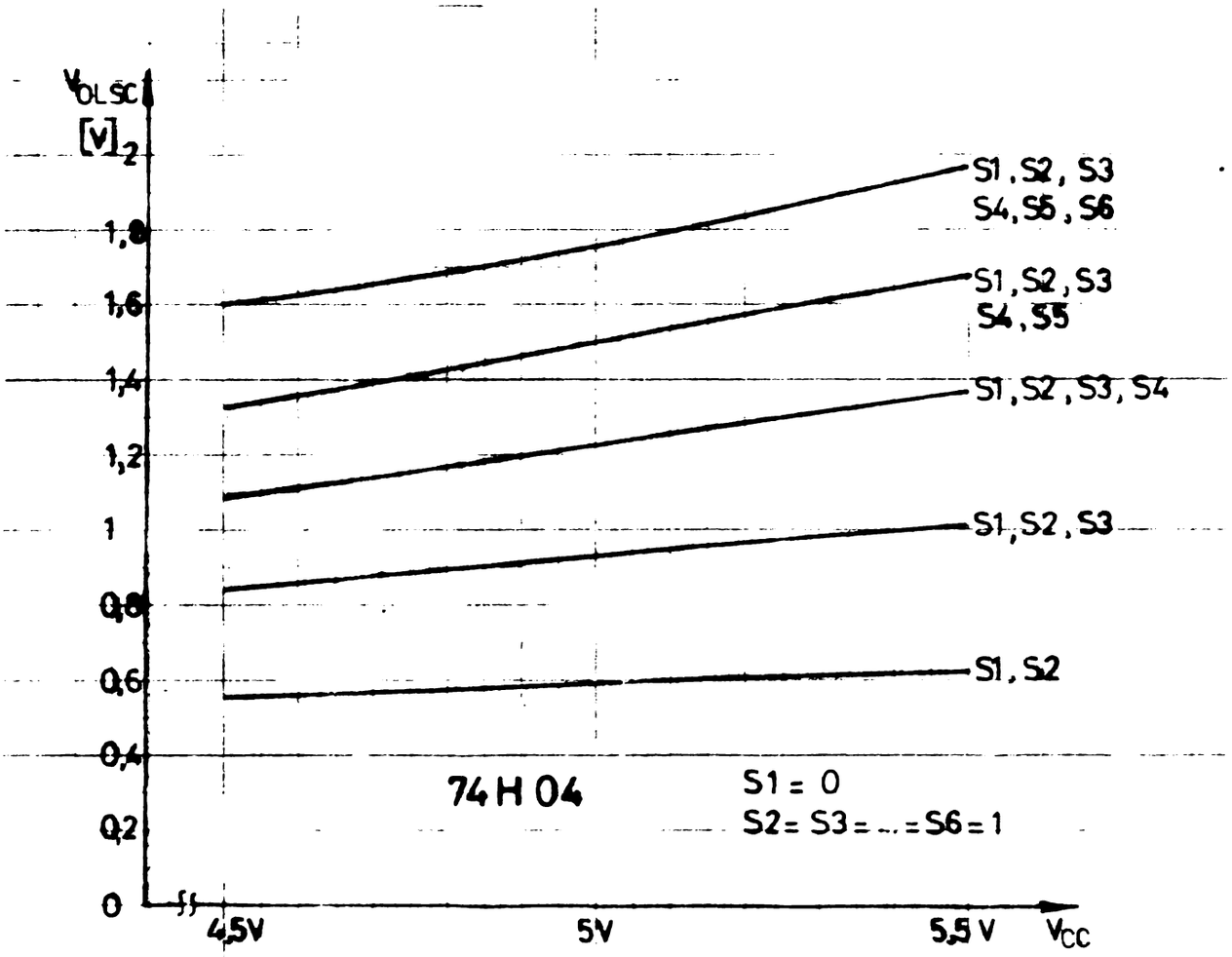


Fig. 4.3. Dependența tensiunii din nodul defect funcție de numărul liniilor în SC, și V_{CC}

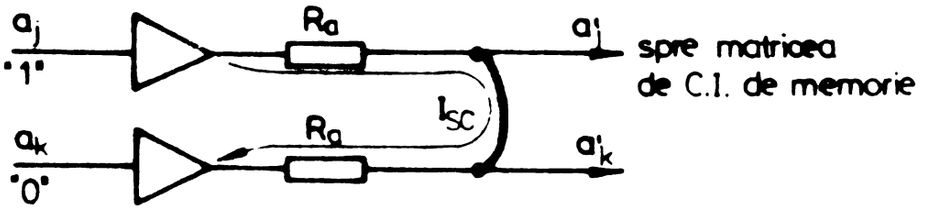


Fig. 4.4. Amplificatoare de adrese cu rezistențe adiționale

evita citirea în cursul testului S3 sau SC a unei locaţii al cărei conţinut este nedefinit autorul recomandă înscrierea unei informaţii fundal D_{2p} în toate locaţiile memoriei testate.

In acest mod, chiar în prezenţa limitărilor de mai sus o linie aflată în scurtcircuit este detectată măcar ca defectă prin blocare dar nu există probabilitatea diagnosticării ca defecte a unor linii care nu sînt defecte.

Experimental s-a constatat că pînă la trei linii în scurtcircuit se realizează funcţia logică SI dacă emiţătoarele sînt neinversoare şi SAU dacă ele sînt inversoare.

Algoritmul SC detectează dar poate diagnostica eronat defecţiuni multiple cu întreruperea traseului de cablaj, de tipul celor din fig.4.5 deoarece s-a presupus că la pinii de

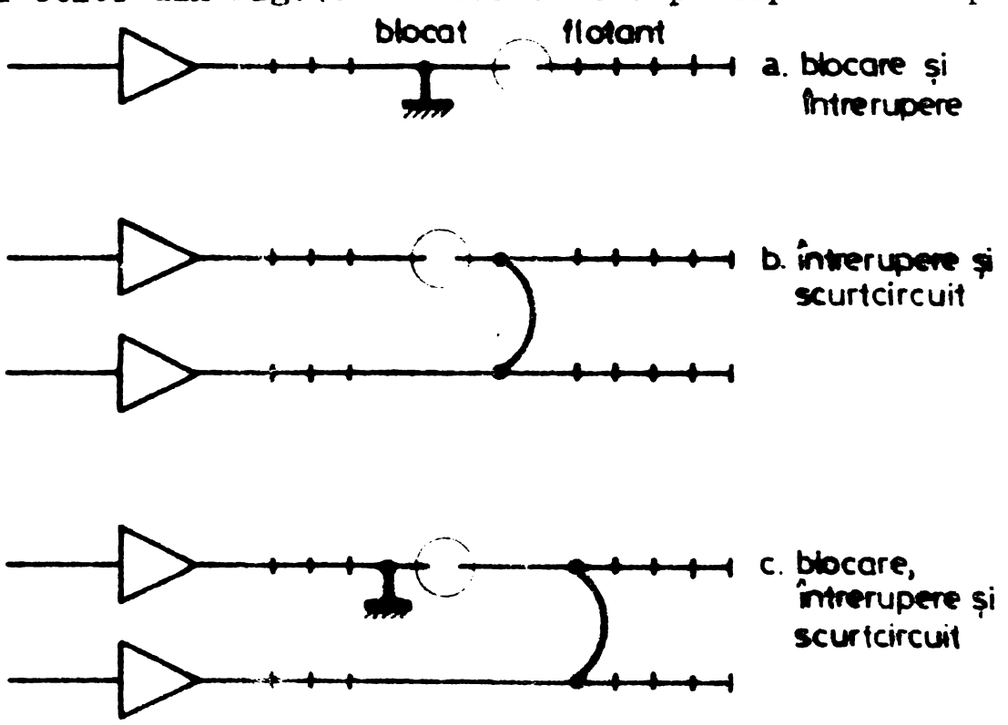


Fig. 4.5. Defecte multiple ce pot fi diagnosticate eronat

adresă de acelaşi rang ai tuturor C.I. de memorie semnalele sînt identice.

Acest tip de defecte multiple poate fi identificat cu uşurinţă prin asociere ci metoda BEM (Board Error Map), prezentată în capitolul 3 sau prin metode de testare neconvenţională [VLAD83].

4.8. Concluzii

Verificarea în sistem a unităţilor de memorie oferă un coeficient de siguranţă destul de mic şi nu poate, în nici un caz, substitui testarea lor pe echipamente specializate. Cu toate acestea, ea poate identifica defecte majore apărute în exploatare sau pe durata transportului. Unele dintre aceste defecte pot fi remediate pe loc, reducîndu-se astfel numărul plachetelor aflate în circulaţie între beneficiari şi producători pentru remedieri.

Cu toate că metodele de testare ale memoriilor şi de evaluare a erorilor sînt cunoscute, programele de autotest care se livrează împreună cu sistemele de calcul nu le utilizează pe cele mai perfecţionate. Utilizarea metodelor perfecţionate de evaluare a erorilor ar duce la o identificare mai rapidă şi mai precisă a defectelor unităţilor de memorie.

In acest capitol au fost prezentate cîteva metode de diagnoză a UM, metode ce pot fi adaptate pentru orice tip de UM cu semiconductoare.

In paragraful 4.5.2 a fost prezentată o generalizare a algoritmului Srini pentru diagnosticarea liniilor de adresă. Utilizînd noul aparat matematic de descriere a defectelor, introdus în § 4.5.1, în paragrafele 4.5.3, 4.5.4 a fost demonstrată capacitatea testului Srini generalizat de identificare a liniilor de adresă defecte atît prin blocare (pentru care a fost prezentat în [SRIN78]) cît şi pentru scurtcircuit.

Au fost găsite două mulţimi de adrese, astfel încît algoritmul Srini generalizat să detecteze liniile de adresă în scurtcircuit, atît în cazul în care acest tip de defecte realizează funcţia logică SAU cablat (dominanţă de 1), cît şi în cazul funcţiei SI cablat (dominanţă de 0). A fost găsită condiţia (necesară şi suficientă) pe care trebuie să o îndeplinească mulţimea adreselor locaţiilor pe care se desfăşoară testele de acest tip pentru ca ele să identifice toate liniile de adresă blocate pe 1 sau 0 (teorema 13). Pînă în prezent, alegerea adreselor pentru teste de acest tip se făcea intuitiv. Au fost demonstrate pentru prima oară teoreme de existenţă şi unicitate pentru mulţimea acestor adrese, astfel încît testele de acest tip să poată identifi-

ca şi liniile de adresă în scurtcircuit, pentru cazurile funcţiilor logice SI şi SAU.

Paragraful 4.6 prezintă un algoritm original denumit SC care permite nu numai identificarea liniilor de adresă defecte ci chiar a defectelor de scurtcircuit, în sensul că este indicat separat fiecare grup de linii în scurtcircuit. Au fost definite condiţiile pe care trebuie să le îndeplinească mulţimea adreselor pe care se desfăşoară testul şi au fost demonstrate posibilităţile acestuia de identificare a defectelor (T1o-T11) Si pentru acest algoritm au fost demonstrate teoreme de existenţă şi unicitate a mulţimilor de adrese ce satisfac condiţiile Cd1 - Cd4, pentru cazurile funcţiilor logice SI şi SAU. (T12 - T15). Algoritmul SC a fost verificat prin simularea defectelor (pe un calculator TRS80, § 4.6.4) şi prin implementarea sa efectivă pe minicalculatorul INDEPENDENT I-100.

Au fost analizate limitele de aplicabilitate ale algoritmilor Srini generalizat şi SC pentru diagnoza liniilor de adresă. Aceste limite se referă la numărul maxim de linii dintr-un grup aflat în scurtcircuit şi se datoresc faptului că la un număr mare de linii nu se mai poate garanta realizarea funcţiei logice SI sau SAU. A fost indicată o soluţie de compromis, care face ca şi în cazul depăşirii limitelor de aplicabilitate, liniile respective să fie indicate ca defecte (chiar dacă se indică defecte de alt tip).

Pentru descrierea defectelor şi demonstrarea celor 15 teoreme a fost utilizat un aparat matematic original.

CAPITOLUL 5.

LIMBAJUL DE TEST MTL

5.1. Limbaje de test - scurt istoric; caracteristici

Nu se poate face un istoric al limbajelor de test fără a aminti, măcar în treacăt, istoricul maşinilor de care acestea sînt legate - echipamentele de testare automată.

In accepţiunea care se dă azi acestui termen, echipamentele de testare automată sînt alcătuite din mai multe dispozitive de stimulare a unităţii supusă testării şi mai multe dispozitive de evaluare (măsurare) a răspunsurilor acesteia, toate fiind conduse de un sistem de calcul (fig.5.1).

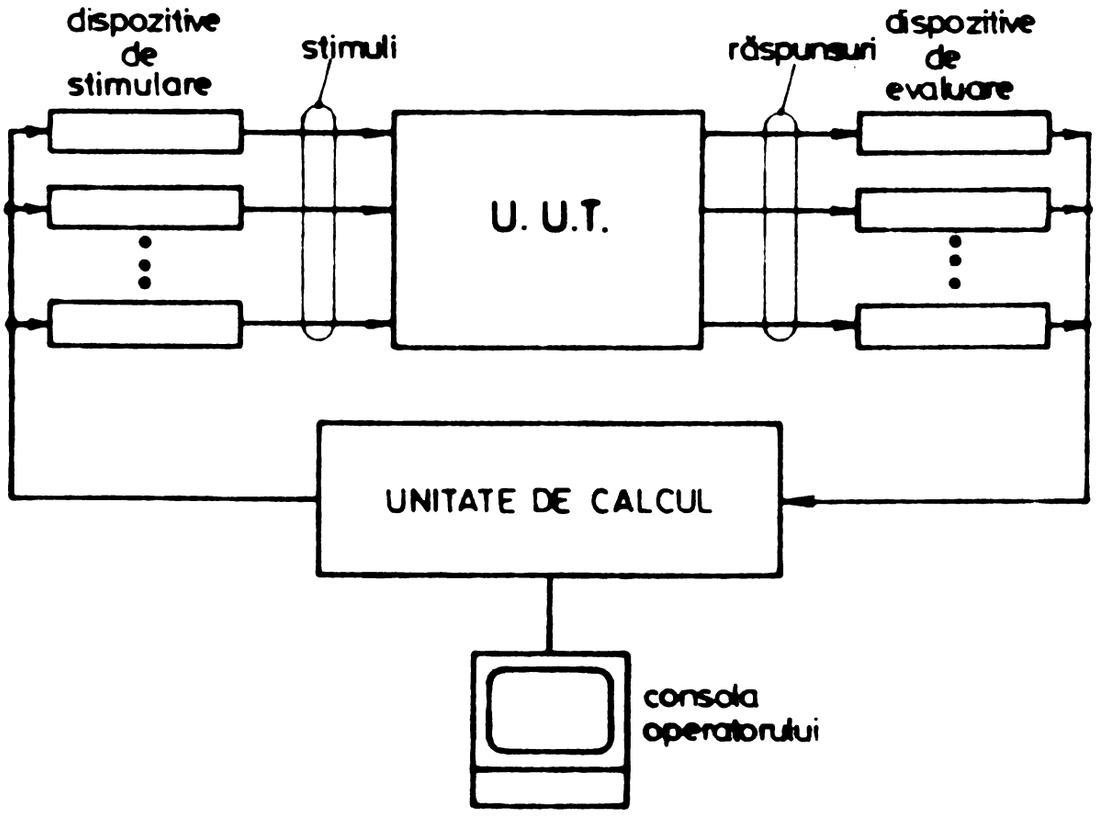


Fig. 5.1. Echipamente de testare automată (schema bloc)

Se consideră la ora actuală [GLAN82a] că primul echipament de testare automată a fost Nortronics DATICO Serial Processor (SP)-5, conceput în 1959 şi livrat începînd din 1963 şi utilizat în testarea subansamblelor electronice ale sistemelor rachete navete Polaris. Performanţele hardware şi software ale acestui echipament par astăzi hilare (memorie cu tambur 4K, programare numai în cod maşină, unitate aritmetică "în majoritate cu tranzistoare"), şi reticenţa clienţilor de a-l utiliza perfect justificată. Oricum sistemul SP-5 a fost un precursor al complexelor echipamente de azi şi implementarea lui a dus la creşterea semnificativă a fiabilităţii sistemelor testate [GLAN82a].

Axioma că fără apariţia limbajelor de nivel înalt nu am fi asistat la expansiunea spectaculoasă a tehnicii de calcul este adevărată şi pentru echipamentele de testare automată.

In accepţiunea cea mai răspîndită, un limbaj de programare este în esenţă un mijloc de reprezentare a algoritmilor [CRET81], un algoritm constînd dintr-o secvenţă finită de operaţii, ordonată şi complet definită, care pornind de la un set de date (intrări) produce un set de date (ieşiri).

Spre deosebire de cazul calculatorului clasic de mai sus, calculatoarele care conduc sisteme de test au următoarele sarcini

a - stimularea dispozitivului testat
b - prelucrarea răspunsurilor dispozitivului testat
c - evaluarea şi sistematizarea acestora
d - producerea setului de date de ieşire (rezultatele testării).

Prin a, b, c ele se aseamănă cu calculatoarele de proces, avînd în plus sarcina producerii setului de date de ieşire. După cum se poate observa instrucţiunile de calcul aritmetic nu sînt neapărat necesare.

Avînd în vedere sarcinile pe care le au de îndeplinit calculatoarele din sistemele de test, limbajele care au apărut pentru acestea fac parte din clasa limbajelor specializate. In continuare se vor face referiri numai la testoarele digitale, cele analogice sau hibride necesitînd în general operaţiuni cu totul diferite.

Un limbaj pentru testarea C.I. logice sau a pachetelor realizate cu acestea, trebuie să posede următoarele tipuri de instrucţiuni.

1. Instrucţiuni pentru alocarea pinilor.

Aceste instrucţiuni descriu conectarea DUT la sistemul de test şi definesc fiecare pin ca alimentare, masă, intrare, ieşire sau intrare/ieşire.

2. Instrucţiuni pentru alocarea nivelurilor logice şi a tensiunilor de alimentare.

Alocarea nivelurilor logice se întîlneşte mai cu seamă în cazul testoarelor de circuite integrate, care utilizează o plachetă separată de stimulare pentru fiecare pin, în esenţă un comutator analogic. In cazul testoarelor de subansamble se utilizează emiţătoare TTL şi nu mai sînt necesare instrucţiunile de stabilire a nivelurilor logice.

3. Instrucţiuni pentru definirea secvenţei de stimulare şi aplicarea ei.

In cazul general definirea se face sub formă de tabel. Este necesară o instrucţiune separată de aplicare, pentru ca aplicarea stimulilor să fie executată simultan pe toţi pini.

4. Instrucţiuni pentru evaluarea răspunsurilor, în general prin comparare cu un răspuns aşteptat, definit printr-o instrucţiune similară celor de tipul 3.

5. Instrucţiuni pentru producerea setului de date la ieşire.

6. Instrucţiuni de salt necondiţionat, condiţionat, apel şi revenire din subrutine.

7. Instrucţiuni de intrare/ieşire.

In cazul general al testării pachetelor logice, calculatorul furnizează dispozitivelor de testare propriu-zise toate cuvintele stimuli ce se vor aplica simultan la un moment dat dispozitivului testat, precum şi toate cuvintele răspuns aşteptate. Aplicarea şi evaluarea poate fi comandată cuvînt cu cuvînt de către calculator (în cazul în care memoria blocurilor de stimulare/evaluare este de un cuvînt) sau pe blocuri (în cazul în care dispozitivul de stimulare/evaluare dispune de o memorie cu dublu acces şi generator de faze). In acest ultim caz este aplicată cu viteză mare o secvenţă de stimulare egală cu numărul de cuvinte din memoria blocului de

stimulare/evaluare.

Aceste memorii, numite şi memorii de pin, sînt în general memorii ECL. Recent au fost realizate sisteme care au lungimea acestei memorii de pînă la 4K cuvinte [WHIT83].

Fluxul de informaţii într-un astfel de testor este prezentat în fig.5.2.

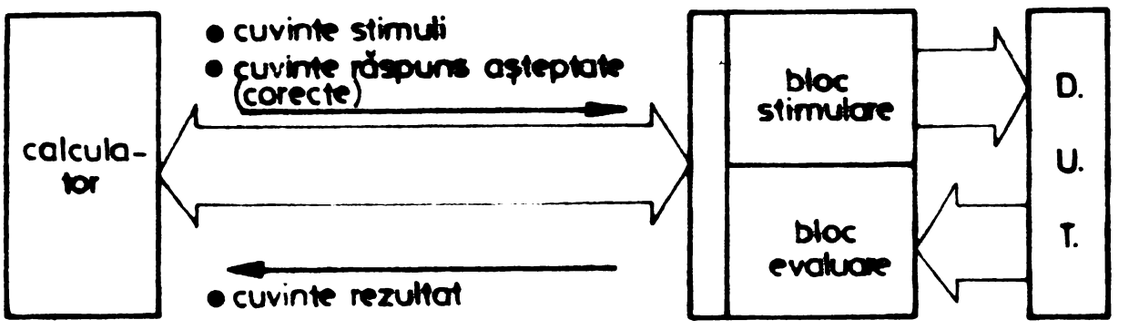


Fig.5.2. Fluxul de informaţii într-un testor universal de plachete logice echipate

Pentru testoare din familia celor de mai sus au fost dezvoltate limbaje de programare de nivel înalt, în general legate de sistemele de test pe care au fost implementate. Aceste limbaje au fost dezvoltate din limbajele FORTRAN sau BASIC sau PASCAL deja existente pentru calculatoarele ce conduceau sistemele de test respective [GLAN82b], [GLAN82d]. Exemple în acest sens le constituie limbajele IDEAL (Marconi-Elliot Avionics Sistem), FLASH (Fault Location and Simulation Hybrid - Membrain), MEMTEST (Membrain) [CATI75], BASCH (Enertec - Schluemberger) [SCHL78a],[SCHL78b] etc. Firma General Radio a dezvoltat un dialect al FORTRAN-ului, introducînd instrucţiuni specifice de testare din clasele 1 - 4 amintite mai sus, şi care sînt descrise în [HART73].

Datorită multitudinii de limbaje de test specializate, la cererea Departamentului Apărării a fost făcută o încercare de a defini un limbaj de test transportabil, independent de resursele sistemului de test, rezultatul fiind limbajul ATLAS (Abbreviated Test Language for Avionics Systems) pus la punct în perioada 1968-1969 de Aeronautic Radio Inc - ARINC [CATI78], [IEEE78].

Insăşi evoluţia acestui limbaj este interesantă şi oglindeşte parţial dezvoltarea echipamentelor de testare automată : definit în 1968 şi aprobat la 4 febr.1969 de Consiliul Administrativ Comun al Liniilor Aeriene suferă 5 modificări oficiale pînă în febr.1972, iar la 31 act.1972 i se adaugă pentru prima oară instrucţiuni pentru testarea echipamentelor digitale.

Este pus apoi în discuţia Comitetului ATLAS al IEEE şi după alte modificări este adoptat ca standard la 1 nov.1976 (Rev.14), sub denumirea de ANSI/IEEE Std.416-1976 [IEEE76]. Ca o recunoaştere a domeniului mai larg de aplicabilitate, a fost schimbată denumirea în "Abbreviated Test Language for **All** Systems".

Nu sînt prezentate în literatură sisteme de test şi compilatoare pentru tot limbajul ATLAS ci numai pentru restricţii sau dialecte ale acestuia.

Cu toate că există un limbaj standard pentru echipamente de testare automată, proliferarea limbajelor de test, derivate sau nu din ATLAS a continuat datorită dificultăţilor de programare în limbajul ATLAS care conţine puţine instrucţiuni referitoare la testarea circuitelor logice.

Limbajul ATLAS rămîne totuşi singurul instrument "care asigură descrierea neambiguă a cerinţelor unei proceduri de test pentru proiectanţi , fabricanţi , utilizatori şi personalul de întreţinere" [IEEE76, paragraf 1.1.2], independent de echipamentul de test utilizat, "fie acesta automatic, manual sau hibrid".

In ţara noastră preocupări în domeniul limbajelor de test au existat în special la I.P.Cluj şi I.P.M. Cluj, unde au fost dezvoltate limbajele LITEST/1, LITEST/2 şi LITEST LSI precum şi translatoarele şi interpretoarele lor, pentru testoare din familia THETA-ROM [CAPA82], [DUMI82], [MIRO82a], [MIRO82b], [NAGY82], [PARA81], [PATA82].

Utilizarea limbajelor cu grad mare de generalitate, din clasa ATLAS, nu este justificată pentru echipamente de test specializate (în această clasă incluzînd şi testoarele de memorii), în primul rînd datorită dificultăţilor de programare. Preţul unui testor de memorii oscilează între 500.000 şi

1.5oo.ooo FF (15o.ooo$- 5oo.ooo$) iar costurile de exploatare în 5 ani (întreţinere şi programare) sînt de patru ori mai mari decît preţul de achiziţie [SCHW81]. Un bun sistem software poate reduce costurile de exploatare cu pînă la 5o% [GLAN82b].

Avînd în vedere cele de mai sus un limbaj eficient trebuie să necesite un efort de programare cît mai redus.

5.2. Limbaje de test pentru testarea memoriilor RAM

Testarea memoriilor, în special a celor cu semiconductoare reprezintă un caz cu totul particular al testării plachetelor logice echipate.

Modul în care se face stimularea şi evaluarea răspunsurilor a fost prezentat în [DANC83a].

In comparaţie cu cazul general al testării (funcţionale) plachetelor logice echipate, testarea memoriilor prezintă următoarele particularităţi:

1. Secvenţele de stimulare sînt foarte lungi (datorită numărului foarte mare de stări).

2. Memoriile trebuie stimulate la frecvenţa maximă de lucru.

Din aceste două motive, definirea şi aplicarea cuvînt cu cuvînt a secvenţei nu este posibilă. Testoarele de memorii dispun în general de un element programabil (procesor rapid) care generează propriu-zis secvenţa de stimulare conform unui program introdus în memoria sa. Instrucţiunile de definire a secvenţei de stimulare constituie de fapt programul rapid.

3. Memoriile au un număr restrîns şi bine definit de tipuri de pini.

a. pini de date de intrare.

b. pini de date de ieşire, în număr egal cu cel al datelor de intrare.

c. pini de adresă (intrări).

d. pini de control (selecţie, scriere, citire etc.), care comandă tipul de ciclu

e. pini de ieşire pentru starea memoriei (semnale de dialog, semnale de control, semnale de la blocurile de detecţie/corecţie a erorilor etc.).

Datorită considerentelor de mai sus şi instrucţiunile de alocare a pinilor pot fi mai simple în cazul memoriilor.

Limbajele de test specializate pentru testoare de memorii trebuie să posede următoarele tipuri de instrucţiuni [DANC82b]:

Instrucţiuni nespecifice testării:

1. Instrucţiuni pentru operaţii I/O şi interfaţă operator.

2. Instrucţiuni de salt condiţionat şi necondiţionat.

3. Apel la subrutine şi revenire din subrutine.

Instrucţiuni specifice testării:

4. Instrucţiuni care stabilesc semnul semnalelor cu care este stimulată memoria testată şi relaţiile de timp dintre ele.

5. Instrucţiuni pentru comanda surselor de alimentare.

6. Instrucţiuni pentru definirea algoritmilor de test (programe ale procesorului rapid).

7. Instrucţiuni pentru execuţia acestor algoritmi (aplicarea secvenţei de stimulare).

8. Instrucţiuni pentru descrierea plăcii de memorie (capacitate, lăţime cuvînt, statică/dinamică, automată/neautomată etc.).

5.3. Metode pentru definirea limbajelor de programare

In definirea sintaxei limbajelor de programare bimba naturală este ambiguă şi redondantă. Din acest motiv, odată cu evoluţia limbajelor pentru calculatoare, a fost necesară introducerea unor metode formale riguroase, care să permită exprimarea corectă, precisă şi concisă a sintaxei acestora. Aceste metode sînt limbaje formale şi se numesc metalimbaje [CRET81].

Prima realizare în acest sens aparţine lui J. Backus, care a definit metalimbajul BNF (Backus Naur Form), utilizat prima dată la descrierea limbajului ALGOL 60. Alt metalimbaj, de asemenea, larg utilizat, este CBL (COBOL-like), utilizat prima oară în descrierea limbajului COBOL.

A treia metodă larg utilizată este cea a diagramelor sintactice.

Se va descrie pe scurt metalimbajul BNF deoarece cu ajutorul acestuia va fi descris limbajul MTL. O descriere mai amănunţită poate fi găsită în [CRET81] şi [AHO77].

Vocabularul metalimbajului BNF este format din metasimboluri, metaconstante, metavariabile şi metaexpresii.

a. Metasimboluri: sînt definite următoarele

::= - metasimbol pentru definire

| - metasimbol pentru alternativă logică avînd semnificaţia "sau"

< > - pentru precizarea metavariabilelor, metavariabilele fiind incluse între paranteze

[ ] - pentru entităţi opţionale

{ } - pentru entităţi care se repetă de un număr de ori, inclusiv zero

b. Metaconstantele sînt elemente ale vocabularului terminal al limbajului care se defineşte.

c. Metavariabilele se obţin prin includerea între paranteze unghiulare a unui şir de caractere reprezentînd o noţiune a limbajului.

d. Metaexpresiile se obţin din metavariabile şi metaconstante unite prin metasimboluri.

5.4. Limbajul MTL

Pentru familia de sisteme de test cu arhitectura ce va fi prezentată în capitolul 6 autorul a definit un limbaj de test, denumit MTL (Memory Test Language), implementabil pe microcalculatoare de 8 biţi. Limbajul de test propus în acest capitol conţine toate tipurile de instrucţiuni enumerate mai sus şi la elaborarea lui s-a avut în vedere în primul rînd reducerea efortului de programare din partea inginerilor de testare.

O linie de program se compune din 4 cîmpuri, despărţite între ele printr-un număr arbitrar de blancuri :

1. Zona etichetă
2. Zona instrucţie
3. Zona operand
4. Zona comentariu

In metaexpresiile de mai jos blancurile vor fi simbolizate prin '␣'.

In metalimbajul BNF o linie de program este definită prin relaţia 5.1.

<linie program> ::=
[<etichetă>][<sep><instrucţie><sep><operand>{<operand>} <sep>]]
[%<comentariu>] (5.1)

<sep> ::= ␣{␣} (5.2)

<comentariu>::= <şir de caractere> (5.3)

<şir de caractere>::= <caracter>{ < caracter >} (5.4)

<caracter> ::=<literă> | < cifră > (5.5)

<cifră>::= 1|2|3|...|9|0 (5.6)

<literă>::= orice caracter tipăribil, avînd cod ISO
şi care nu este cifră, inclusiv semne
de punctuaţie (5.7)

<instrucţie>::=<instrucţiune TITLE>|<instrucţiune NAME>|
<ins. PRINT>|<ins. BEEP>|<ins. BEM>|<ins. WAIT>|<ins. OPI>|
<ins.STP>|<ins.END>|<ins.GTO>|<ins.CALL>|<ins.RETURN>|
<ins.IFKEY>|<ins.IFERROR>|<ins.TIMING>|<ins.PWS>|<ins.FPP>|
<ins.COD>|<ins.FPWAIT>|<ins.SET>|<ins.TOPO>|<ins.DFM>|
<ins.INTEL> (5.8)

Metavariabilele <instrucţie >, <operand> vor fi definite în subcapitolele pentru descrierea instrucţiunilor limbajului.

Etichetele sînt utilizate pentru identificarea liniilor de program în cadrul instrucţiunilor de salt.

Orice şir de caractere care începe din prima coloană şi respectă sintaxa definită prin metaexpresiile (5.7), (5.8) este interpretat ca etichetă. Primul caracter din şir trebuie să fie o literă majusculă.

<etichetă>::=<majusculă> { < caracter > } (5.9)

<majusculă>::= A|B|C|...|X|Y|Z (5.1o)

Sînt interpretate numai primele 6 caractere ale etichetei. De exemplu

LABEL1
LABEL1O
LABEL16

vor fi interpretate ca etichetă multidefinită.

Instrucţiunile, în număr de 23, sînt recunoscute după primele 3 caractere

PRINT
PRIN semnifică aceeaşi instrucţiune
PRI

o instrucţiune poate fi continuată pe linia următoare dacă se utilizează semnul ; .

Zona comentariu poate începe din orice coloană, inclusiv prima, în cazul în care linia de program conţine numai un comentariu.

Numărul operanzilor şi sintaxa lor depinde de instrucţia respectivă.

5.4.1. Grupa instrucţiunilor de I/O şi interfaţa operator

a. Instrucţiunea TITLE

Sintaxă:

<instrucţiunea TITLE> ::=

TITLE <sep> <operand TITLE> (5.11)

<operand TITLE>::= "<nume program>" (5.12)

<nume program>::= <şir de caractere> (5.13)

Descriere :

Este o instrucţiune utilizată pentru a marca începutul programului. Are ca efect tipărirea la consolă şi la imprimantă (în cazul în care aceasta nu este asignată ca şi consola) a mesajului:

START OF <nume program> ON <serie>

unde <serie> este un şir de caractere introduse

<serie> ::= <şir de caractere> (5.14)

de operator prin comanda SERIE înainte de începerea rulării, de obicei seria de identificare a plăcii testate.

b. Instrucţiunea NAME

Sintaxa

<instr.NAME> ::= NAME <sep> <operand NAME> (5.15)

<operand NAME> ::= "<bloc teste>" (5.16)

<bloc teste> ::= <şir de caractere> (5.17)

Descriere:

Este o instrucţiune care introduce şirul de caractere dintre " " într-un buffer intern şi, simultan îl tipăreşte la consolă. Această tipărire poate fi inhibată de operator printr-o comandă PRINT N.

c. Instrucţiunea PRINT

Sintaxă

<instr.PRINT> ::= PRINT <sep><operand PRINT>[<sep>/] (5.18)

<operand PRINT> :: " <text> " (5.19)

<text> ::= <şir de caractere> (5.20)

Este o instrucţiune de tipărire la consolă, cu sau fără trecere la rînd nou după tipărire, în funcţie de prezenţa - absenţa parametrului / .

d. Instrucţiunea BEEP

Sintaxă

<instrucţiunea BEEP> ::= BEEP (5.21)

Este o instrucţiune de avertizare acustică.

e. Instrucţiunea BEM

Sintaxă

<instrucţiunea BEM>::= BEM (5.22)

Are ca efect tipărirea/afişarea hărţii erorilor pe placa de memorie testată (Board Error Map).

f. Instrucţiunea WAIT

Sintaxă

<instr.WAIT>::= WAIT [<sep><operand WAIT>] (5.23)

<operand WAIT>::= <număr întreg> (5.24)

<număr întreg>::= <cifră>{<cifră>} (5.25)

Această instrucţiune are ca efect introducerea unei pauze. Durata acesteia, exprimată ca un număr întreg de milisecunde, între 1 -9999 este dată de operand. In cazul în care operandul lipseşte, pauza introdusă este de 1 ms.

g. Instrucţiunea OPI

Sintaxă

<instr.OPI>::= OPI (5.26)

Această instrucţiune suspendă execuţia programului. Relansarea se face numai după comanda REL sau GO dată de operator. Operatorul este avertizat prin mesajul "INVI" că se aşteaptă o intervenţie din partea sa. Este utilizată pentru suspendarea execuţiei programului în vederea executării unor manevre.

Exemplu

```
%  Secvenţă de program pentru reglaj
   NAME "REGLAJ TACT DATE IESIRE"
   PRINT "CONECTATI SONDA LA TP23" /
   OPI % Aşteaptă execuţie manevră şi comandă continuare
   COD CK I % Lansare secvenţă de stimulare cu durată
            % nedeterminată
   PRINT "REGLATI LA 310 NS" /
   OPI % Aşteptare execuţie reglaj şi comandă continuare
   GTO TEST % Salt în programul de testare.
```

h. Instrucţiunea STP

Sintaxă

<instr.STP>::= STP (5.27)

Cu această instrucţiune programatorul poate introduce puncte de oprire în program în scopul verificării şi punerii la punct a acestuia. Este tipărită ultima linie de program şi numărul ei.

i. Instrucţiunea END

Sintaxă

<instr.END> ::= END (5.28)

Această instrucţiune are ca efect încheierea execuţiei programului de test şi afişarea rezultatului sub forma:

END OF <nume program> ON <serie>

REZULT : <rezultat>

unde

<rezultat>::= PASS|FAIL

Tipărirea se face la consolă şi la imprimantă dacă aceasta nu este asigurată ca şi consolă.

<nume program> şi <serie> sînt definite prin instrucţia TITLE respectiv comanda SERIE.

5.4.2. Grupa instrucţiunilor de salt, apel la subrutine

Instrucţiuni de salt şi apel necondiţionat:

<instr.GTO>::= GTO < sep >< etichetă > (5.29)

<instr.CALL>::= CALL < sep >< etichetă > (5.3o)

<instr.RETURN>::= RETURN (5.31)

Instrucţiuni de salt condiţionat:

<instr.IFKEY>::= IFKEY <sep> <etichetă> (5.32)

<instr.IFERROR>::= IFERR <sep> <etichetă> (5.33)

Prima instrucţiune de salt condiţionat este utilizată pentru a permite operatorului să execute numai anumite secvenţe din programul de test.

Exemplu:

```
START
     PRINT "PENTRU REGLAJE PUNETI CHEIA K1"
     OPI
     IFKEY REGLAJE
     PRINT "PENTRU TEST FINAL PUNETI CHEIA K1"
     IFKEY TESTFIN
     PRINT "EROARE DE OPERARE"
     BEEP
     WAIT 100
     BEEP
     GTO START
```

Instrucţiunea IFERROR permite ramificarea programului funcţiei de rezultatul ultimului test executat. Execuţia saltului este validată de poziţia unei chei de pe panoul testorului.

5.4.3. Instrucţiunea TIMING

Este utilizată pentru definirea relaţiilor temporale dintre semnale, precum şi a polarităţii acestora (normal - complementar).

Sintaxa:

⟨instrucţiunea TIMING⟩::=
TIMING⟨sep⟩⟨ident.semnal⟩=⟨operand TIMING⟩$\left\{, \langle\text{operand TIMING}\rangle\right\}_2^{\bar{2}}$
[, ⟨semn⟩] (5.34)

⟨ident.semnal⟩::= TD|TT|TR|TM|TO|TI|TA|TE|TS|TW|TC|TB|TG
[⟨şir de caractere⟩] (5.35)

⟨operand TIMING⟩::= ⟨număr întreg⟩ (5.36)

⟨semn⟩ ::= +|− (5.37)

Descriere:

Semnalul la care se referă instrucţiunea este identificat numai prin un singur caracter, însă pentru uşurinţa programării se poate scrie denumirea sa completă.

Exemplu:

TIMING TWRITE = 0100, 0200, 0150,-

Există doi identificatori TT, TR care nu se referă la un semnal anume ci la perioada de repetiție a ciclurilor de acces respectiv regenerare.

Se pot programa pozițiile temporale ale fronturilor (anterioare și posterioare) a 11 semnale: date de intrare, adrese, strob date de ieșire (timp acces), scriere, cerere de ciclu, selecție etc. Numărul nn exprimă intervalele de timp în nanosecunde, respectiv microsecunde pentru regenerare.

5.4.4. Grupa instrucțiunilor de definire și aplicare stimuli

a. Comanda surselor de alimentare se face prin instrucțiunea PWSUPPLY care are sintaxa generală

⟨instrucțiunea PWSUPPLY⟩::=
PWSUPPLY ⟨sep⟩ ⟨identificator sursă⟩ = ⟨număr real⟩ (5.38)

⟨identificator sursă⟩::= U1 | U2 | U3 (5.39)

⟨număr real⟩::=[⟨semn⟩]{⟨cifră⟩}[.{⟨cifră⟩}] (5.40)

b. Definirea secvenței de stimulare se face prin instrucțiunea FPP (Fast Processor Program).

⟨instrucțiune definire secvență⟩::=
FPP ⟨sep⟩ ⟨tip cod⟩␣⟨tp.adr⟩␣⟨program proc.rapid⟩ (5.41)

⟨tip cod⟩::= 00 | FF (5.42)
⟨tp.adr⟩::= ⟨număr hexa⟩ (5.43)
⟨număr hexa⟩::= ⟨cifră hexa⟩{⟨cifră hexa⟩} (5.44)
⟨cifră hexa⟩::= ⟨cifră⟩ | A | B | C | D | E | F (5.45)
⟨program proc.rapid⟩::= ⟨șir hexa⟩ (5.46)
⟨șir hexa⟩::={⟨număr hexa⟩␣} (5.47)

Relațiile (5.41) - (5.47) definesc complet sintaxa instrucțiunii. Semnificația variabilelor ⟨tip cod⟩, ⟨tp.adr⟩ i.e. adresa subrutinei de terminare parțială, și ⟨program procesor rapid⟩ vor fi prezentate în capitolul 6.

c. Execuția (lansarea) testelor se face prin instrucțiunea COD.

⟨instr.exec.secvență⟩::=
COD ⟨sep⟩ ⟨nume COD⟩[⟨sep⟩ I] (5.48)

<nume cod>::= X?|CK|PARITY|GALROW|GWR|MASEST|EXDATA|WALK|MARCH|
INDATA|GALCOL|GALDIA|GALPAT|REFRST|REFRDN|
SOCTET|FBAT|TRIST|ERRPAR|AFPX|AFPY|SELMEM
(5.49)

In interpretor sînt incorporate programele ale procesorului rapid coresounzătoare la 21 teste standard, care pot fi executate direct, fără a fi definite anterior prin instrucţiunea FPP.

Parametrul opţional I semnifică lansarea procesorului rapid în regim de test infinit, caz în care nu se face interpretarea rezultatului şi, imediat după lansare se trece la execuţia următoarei linii de program.

S-a ales varianta cu încorporarea în interpretor a majorităţii testelor standard deoarece în acest fel lungimea unui program se reduce în proporţie de 5o% - 8o% iar efortul de programare este de asemenea mai redus.

Lansarea în regim infinit a procesorului rapid permite rularea în continuare a programului în timp ce placa testată este stimulată ciclic pentru execuţia unor reglaje sau măsurători. Deoarece, în cazul lansăriicodurilor în regim infinit executivul nu aşteaptă terminarea secvenţei ci trece la linia de program următoare, dacă se lansează un cod în regim infinit în scopul vizualizării unor semnale din UM, instrucţia COD I trebuie urmată de o instrucţiune care să oprească rularea programului pînă la intervenţia operatorului (OPI, STP, IFKEY). Un exemplu de utilizare ale acestei instrucţiuni a fost dat în § 5.3.1.

Codul definit prin instrucţiunea FPP este lansat sub numele X?

COD X?

d. Unii algoritmi necesită <u>temporizări</u> (pauze) ale procesorului rapid, de valoare mare, utilizate în general pentru verificarea capacităţii de menţinere a informaţiei în absenţa stimulării.

Durata acestor temporizări este fixată prin instrucţia FPWAIT

<instrucţiune FPWAIT>::= FPWAIT[<sep><operand WAIT>] (5.5o)

unde <operand WAIT> , este o variabilă cu sintaxa definită prin relaţia (5.24) şi exprimă temporizarea în milisecunde.

5.4.5. Instrucţiuni pentru descrierea unităţii de memorie testate

Instrucţiunea principală pentru descrierea unităţii de memorie testate este DFM (define memory), cu sintaxa:

<instrucţie DFM>::= DFM <sep> < parametrii> (5.51)

<parametrii>::= <şir hexa> (5.52)

Prin această instrucţiune se definesc:

- masca pentru căile de date testate
- capacitatea unităţii de memorie pe cele trei axe (X,Y şi Z)
- tipul căilor de date (magistrale bi- sau unidirecţionale)
- tipul unităţii statică/dinamică, autonomă/neautonomă.

Parametrii sînt identificaţi după poziţia lor în şir. Semnificaţia fiecărui parametru este descrisă în [TEL82a].

Pentru definirea topologiei plăcii testate în vederea adresării topologice [DANC83a], se utilizează instrucţiunea TOPO.

Această instrucţiune serveşte atît pentru definirea topologiei cît şi pentru stabilirea modului de adresare topologic sau netopologic.

Sintaxa:

<instrucţiune TOPO>::

TOPO <sep> <comandă> |<id.axă><sep><topologie> (5.53)

<comandă>::= SET|RESET (5.54)

<id.axă>::= X|Y|Z (5.55)

<topologie>::= N | <şir hexa> (5.56)

Dacă nu se doreşte adresarea topologică pe toate axele ci numai pe una sau două dintre ele, se poate evita scrierea, definirea topologiei pe acestea, utilizînd caracterul N în loc de parametrii topologiei.

Stabilirea modului de adresare se face prin:

TOPO SET % adresare topologică

TOPO RESET % adresare netopologică

Ex. Secvenţă de program cu utilizarea instrucţiunii TOPO :

```
% Definirea topologiei C.I. de memorie
% Adresare topologică pe X şi Y, netopologică pe Z
%
  TOPO   X   <topologie>
  TOPO   Y   <topologie>
  TOPO   Z    N
% Stabilirea modului de adresare topologică
  TOPO   SET
  NAME   "TOPOLOGIC"
  CALL   TEST1    % Apel la subrutina ce execută
                    testele
  BEM             % Tipărire Board Error Map
% Restabilirea modului de adresare netopologic
  TOPO   RESET
  NAME   "NETOPOLOGIC"
  CALL   TEST2
  BEM             % Tipărire Board Error MAP
% Restabilirea modului de adresare topologic
  TOPO   SET
  NAME   "T3-TOPO"
  CALL   TEST3
  BEM
  END
```

5.4.6. Instrucţiuni speciale

In această categorie am inclus acele instrucţiuni care permit execuţia unor programe/secvenţe de program/ scrise în cod maşină.

a. Instrucţiunea SET permite definirea conţinutului unei zone din memoria microcalculatorului, începînd de la adresă .

Ea are sintaxa :

<instrucţiune SET>::= SET <sep> <adresă> <sep> <octeţi> (5.57)

<adresă>::= <număr hexa> (5.58)

<octeţi>::= <şir hexa> (5.59)

b. Instrucţiunea INTEL

<instrucţiunea INTEL>::=

INTEL <sep > < adresă> (5.6o)

permite apelul la o subrutină scrisă în cod maşină (eventual introdusă în memorie prin SET) cu adresa precizată în zona operand.

5.5. Concluzii asupra limbajului MTL

Limbajul MTL definit în acest capitol este implementabil pe microcalculatoare bazate pe procesoare de 8 biţi de tip 8o8o/8o85.

Limbajul MTL conţine toate instrucţiunile necesare oricărui tip de stimulare/evaluare a unei unităţi de memorie.

Este un limbaj orientat spre utilizator spre deosebire de limbajul ATLAS şi cele derivate din acesta care sînt orientate spre dispozitivul testat ("Unit Under Test Orientation", § 1,1.1 în [IEEE7o]).

Oferă operatorului posibilitatea de a influenţa desfăşurarea testării prin intervenţii minime şi numai de la panoul de comandă al sistemului de test. Pentru comparaţie, alte limbaje permit acest lucru numai prin intervenţii de la consolă .

Pentru a reduce efortul de programare la un minim, sintaxa instrucţiunilor a fost în mod voit simplificată, unele dintre instrucţiuni fiind în fapt macro instrucţiuni, ce cuprind atît definirea cît şi aplicarea stimulilor şi evaluarea răspunsurilor. De exemplu, pentru implementarea instrucţiuni COD AFFX, ar fi necesare peste 25 instrucţiuni în limbajul ATLAS sau dialectele sale. Se obţine astfel o reducere a lungimii programelor (şi a efortului de programare) cu 5o-8o %.

In anexa A5.e este prezentat programul pentru testarea modulelor de memorie tip MMIlo2, cu o capacitate de 128 Kocteţi.

CAPITOLUL 6.

SOFTWARE DE BAZA PENTRU UN ECHIPAMENT DE TESTARE AUTOMATA A UNITATILOR DE MEMORIE

6.1. Introducere

Primul sistem de testare automată a memoriilor, condus de un calculator, a fost prezentat în anul 1976 [ADAR8o], de către firma Adar Associates, astăzi Adar-Scientific Atlanta. De altfel, această firmă se menţine şi azi ca leader, în privinţa metodelor de testare a memoriilor, chiar dacă din punct de vedere al performanţelor, testoarele sale sînt surclasate de cele ale marilor firme ca Tektronix, GenRad, Teradyne sau Takeda Riken. Sistemul de test, prezentat de Adar în 1976, era condus de un minicalculator de 16 biţi şi avea ca memorie externă numai bandă perforată şi casetă magnetică. De altfel, caracteristic tuturor sistemelor apărute ulterior este faptul că ele sînt conduse de minicalculatoare de 16 biţi, în general din clasa PDP-11, datorită uriaşelor resurse software existente pentru aceste calculatoare. Sînt menţionate, în literatură, chiar şi sisteme de test conduse de minicalculatoare de 32 bit (ex. Bendix 32o [GALN82b]).

Evoluţia echipamentelor de testare a memoriilor MOS realizate la ITC Timişoara, a urmat aceeaşi linie ca şi cea a evoluţiei acestora pe plan mondial [RIMB82].

Pornind de la observaţia că timpul testării este ocupat în cea mai mare parte de execuţia testelor propriu-zise, sau o operaţie de intrare/ieşire şi nu de prelucrarea rezultatelor testelor, la ITC Timişoara a fost realizată o familie de astfel de testoare, grupate în jurul unui microcalculator bazat pe microprocesorul 8o85. Chiar dacă acesta are o viteză de calcul mult mai mică decît minicalculatoarele de 16 sau 32

biţi, timpul necesar testării nu creşte semnificativ prin utilizarea unui microprocesor de 8 biţi.

Prezentarea sistemului software dezvoltat de autor, pentru familia de echipamente de testare automată a UM, MOSTEST-03, realizate la ITC filiala Timişoara, constituie subiectul acestui capitol.

Programul, în limbaj de asamblare, (anexa A6) este alcătuit din cca 6900 linii sursă şi se întinde pe cca 120 pagini. Deoarece descrierea funcţionării unui program este în general mai întinsă decît programul propriu-zis, autorul a preferat ca în acest capitol să prezinte numai structura şi funcţionarea la nivel macro a sistemului software. Vor fi prezentate, numai ca exemple şi fără amănunte descriptive, numai două dintre subrutine.

Este prezentată varianta 2.1 a sistemului software MOSTEST-03D. Un echipament de testare automată a memoriilor, cu această variantă de sistem software, se află în exploatare la FMECTC Timişoara de peste un an, cu rezultate deosebite.

6.2. Structura generală a sistemelor de test din familia MOSTEST-03

6.2.1. Structura sistemului de test

Sistemele de test din această familie sînt organizate conform structurii prezentate în fig.6.1, [DANC82b].

Toate elementele sistemului sînt grupate în jurul unui microcalculator, bazat pe procesorul I-8085, avînd ca echipamente periferice principale o consolă serială cu imprimantă, un display rapid şi o unitate de disc flexibil.

Procesorul rapid programabil este elementul principal din sistem, datorită faptului că generează de fapt secvenţa de stimulare a UM.

El conţine o memorie de microprogram cu 16 cuvinte a 40 biţi, o memorie topologică, o memorie pentru realizarea Board Error Map (BEM), precum şi numărătoarele de adrese şi registrele de comparaţie. Lăţimea cuvîntului de date ce poate fi generat este de 24 biţi în două grupe a cîte 12 biţi [BALM82].

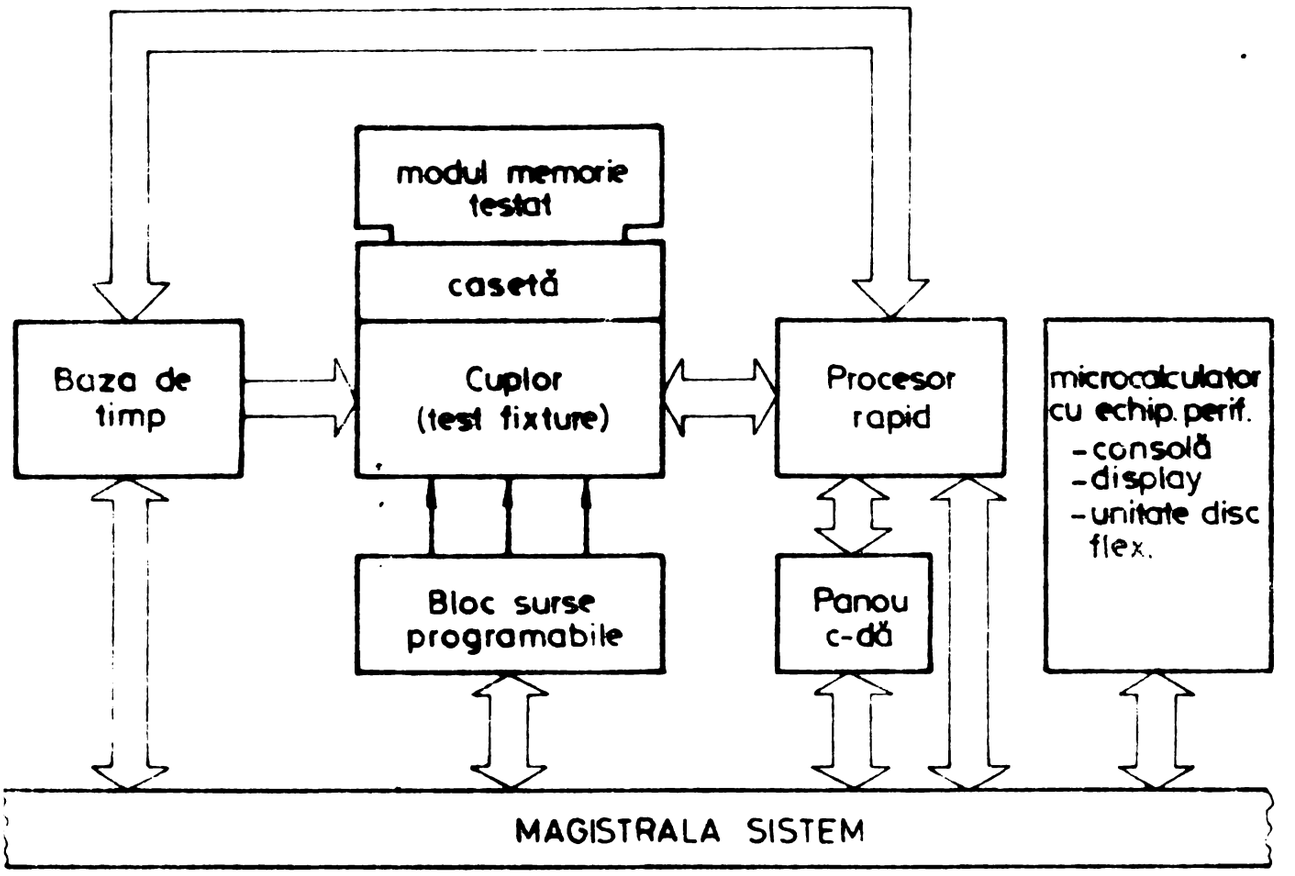


Fig. 6.1. Structura sistemului de test

Baza de timp asigură polaritatea şi relaţiile de timp între semnalele cu care este stimulată unitatea testată. Poate genera 11 semnale cu 1, 2 sau 3 fronturi programabile, cu o rezoluţie de 5 ns [GHER82] .

Blocul surselor programabile asigură alimentarea unităţii testate; ele pot genera tensiunile de alimentare cu o rezoluţie de 20 mV.

Cuplorul (test fixture) este elementul care asigură formarea propriu-zisă a semnalelor de stimulare, precum şi strobarea ieşirilor unităţii testate. Este de asemenea programabil.

Panoul de comandă conţine un număr de 6 chei, văzute ca port de intrare de către microcalculator, 3 chei pentru fortarea anumitor regimuri ale procesorului rapid, precum şi 3

te:ie legate la sistemul de întreruperi.

Microcalculatorul acceptă din partea blocurilor funcţi:nale următoarele întreruperi

- de la panoul de comandă START, RELUARE, STOP
- de la procesorul rapid -terminare secvenţă test
  -cerere de temporizare
- de la cuplor şi baza de timp semnalizare defect major, i.e. depăşirea curenţilor de alimentare admişi şi avarie în baza de timp (tip watch-dog).

6.2.2. Structura microcalculatorului

In fig.6.2 este prezentată configuraţia microcalculatoruluicare conduce sistemul de test [DANC82a], [TELE82b].

El este alcătuit dintr-o unitate centrală Intel 8085, avînd ca memorie 4 KO EPROM şi 32 KO RAM dinamic.

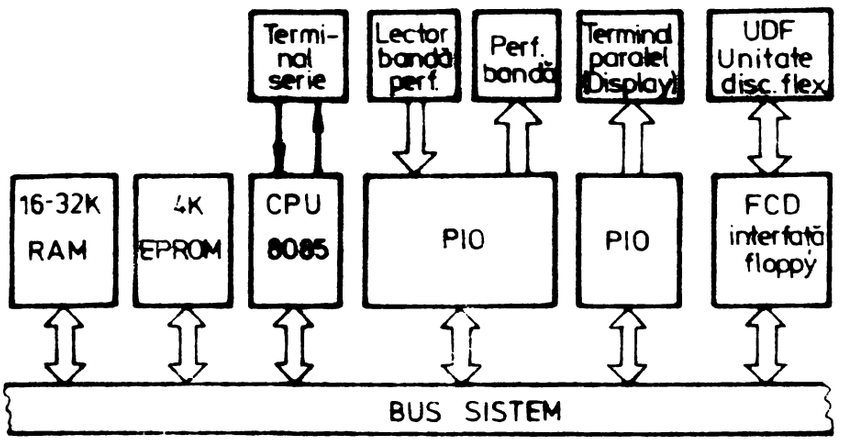


Fig.6.2. Structura microcalculatorului

Memoria de masă este asigurată de o unitate de floppy disc. Există de asemenea interfeţe paralele de intrare/ieşire, pentru lucrul cu bandă perforată. Datorită dificultăţilor şi dezavantajelor benzii perforate, acestea nu sînt utilizate decît în cazul unor defecţiuni majore ale memoriei de masă cu disc flexibil.

Pentru interfaţa cu operatorul au fost prevăzute două periferice: o consolă serie de intrare/ieşire şi un display paralel [DANC82a], [TELE82b]. A fost aleasă această din urmă soluţie pentru afişarea rezultatelor, datorită vitezei supe-

rioare de vizualizare a datelor. Mai cu seamă în etapa de punere la punct şi depanare, este necesară vizualizarea foarte frecventă a hărţii erorilor pe placă - BEM (Board Error Map). Aceasta comportă pînă la 4oo caractere, durata transferului serie cu un periferic lent, (3oo baud), fiind mult mai mare decît durata execuţiei testelor.

6.3. Sistem software pentru echipamente de testare automată a unităţilor de memorie

6.3.1. Necesităţi software

Se consideră că la ora actuală dotarea software constituie 9o-95% din preţul unui calculator de uz general. Pentru echipamente de testare automată acest procent este mult mai redus datorită preţului, în general foarte ridicat, al aparaturii complexe de stimulare/evaluare, dar tendinţa sa de creştere este evidentă în ultimii ani. Ea este justificată de faptul că, în costul de exploatare pe 5 ani al unui echipament de testare automată, preţul de achiziţie reprezintă numai 2o%, restul de 8o% fiind constituit de cheltuielile de întreţinere şi mai ales, de dezvoltare de programe de test [SCHW81]. După cum s-a menţionat deja în capitolul precedent, un sistem software performant poate reduce costurile de exploatare cu pînă la 5o% [GLAN82b].

Cerinţele pentru un sistem software care fac ca un MTA să fie eficient din punct de vedere economic sînt:

- modul de operare, inclusiv pornirea sistemului, trebuie să fie cît mai simplu, pentru a permite operarea de către personal cu calificare medie sau mică,
- interpretarea şi sistematizarea erorilor să fie făcută în cea mai mare parte de către calculator (depanare asistată de calculator), cu un efort minim din partea programatorului. Acest lucru înseamnă implementarea direct în executiv a acestor facilităţi (sau a unei părţi cît mai mari),
- este de dorit ca redactarea şi verificarea programelor să se facă iterativ, pentru a reduce efortul de programare,
- tratarea erorilor de programare şi manevrare să fie

astfel făcută încît să nu ducă la distrugerea programului de test încărcat şi nici,(dacă este posibil), la pierderea rezultatelor testelor executate pînă în acel moment.

Ca şi orice sistem de calcul, un echipament de testare automată necesită:

- un sistem de operare, pentru gestionarea programelor şi a lansării sarcinilor,
- un editor de text,pentru redactarea şi întreţinerea programelor sursă,
- un executiv, sub controlul căruia se efectuează rularea programelor de test scrise într-un limbaj de nivel înalt

Pentru sistemul de operare se acceptă în general definiţia propusă de Creech şi reluată de Randell şi [BALT74]:

"Sistemul de operare este acea parte a sistemului de calcul care asigură alocarea şi coordonarea resurselor sistemului, precum şi o serie de funcţiuni pentru asistarea programatorilor, cu obiectivul de a asigura performanţele optime ale sistemului".

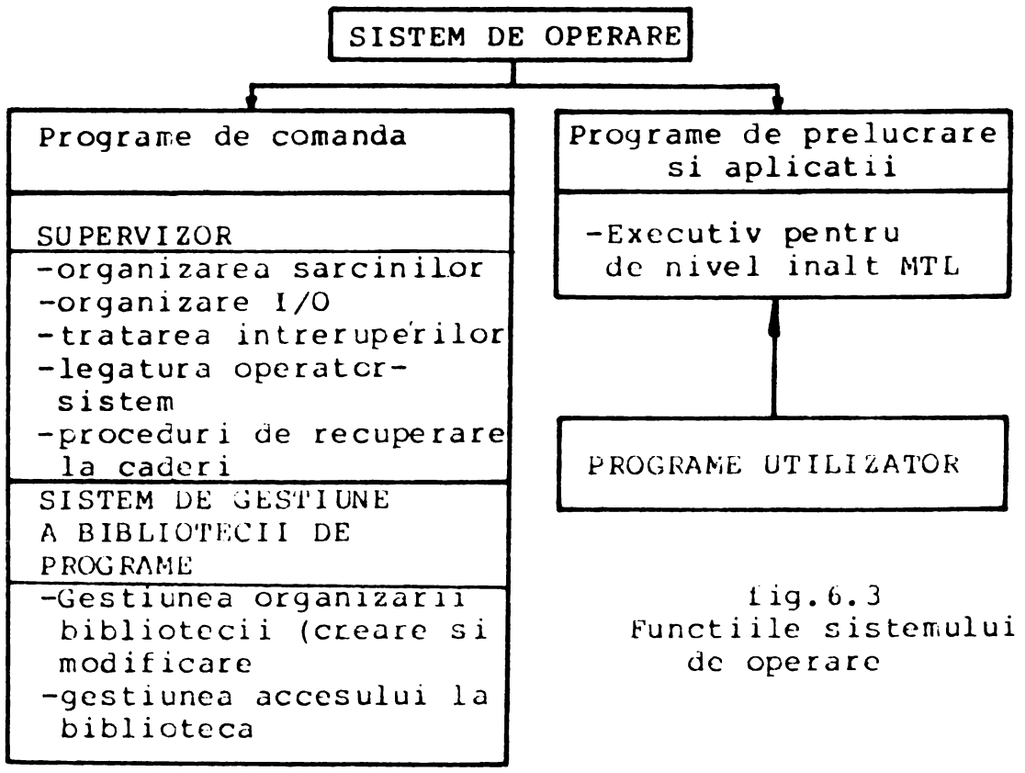


fig.6.3
Functiile sistemului de operare

Dintre funcţiile sistemelor de operare sistematizate de [BALT74], autorul a reţinut,pentru alcătuirea sistemului de operare MICRODOS,cele din fig.6.3. S-a putut renunţa la o

parte din funcţiile prezentate de [BALT74], în primul rînd pentru că sistemul de operare MICRODOS este destinat unui microcalculator cu o aplicaţie unică. In al doilea rînd, existînd întotdeauna o singură lucrare în execuţie, nu sînt necesare funcţii specifice sistemelor multi-task cum sînt: organizarea lucrărilor şi a sarcinilor, alocarea memoriei, înlănţuirea segmentelor, contabilitate, generare sistem etc.

6.3.2. Structură generală software

Arhitectura sistemului software conceput de autor, pentru familia de echipamente de testare automată a memoriilor LOSTEST-03, şi realizat în colaborare cu un colectiv de la ITC filiala Timişoara, este prezentată în fig.6.4. El este alcătuit din patru elemente de bază DEBUGGER, MICRODOS, EDITOR şi TEST INTERPRETER. In figură sînt prezentate comenzile, în urma cărora se transferă controlul între cele patru blocuri software.

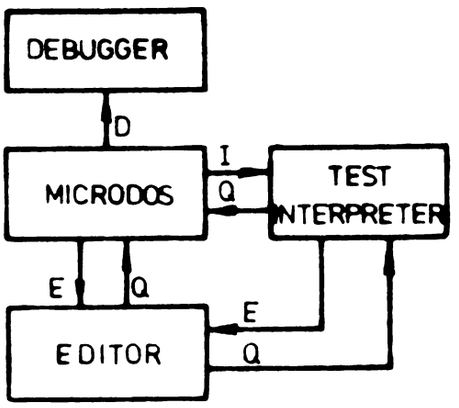


Fig.6.4. Structura software

Nucleul sistemului software îl constituie sistemul de operare MICRODOS, rezident în RAM.

La pornirea sistemului, datorită unui BOOTSTRAP LOADER rezident în memoria EPROM, sistemul de operare MICRODOS (MICRO Disk Operating System) este încărcat şi lansat în execuţie.

El permite accesul în programele DEBUGGER, rezident în EPROM, EDITOR rezident, şi TEST INTERPRETOR, executivul pentru limbajul MTL.

Pentru simplificarea manevrelor, a reducerii duratei necesare modificării unui program de test, precum şi a reducerii probabilităţii manevrelor greşite, accesul în EDITOR este posibil şi din INTERPRETOR, revenirea din EDITOR efectuîndu-se întotdeauna în programul apelant.

Se va face în continuare o scurtă prezentare a sistemului de operare şi a editorului. O descriere mai amănunţită poate fi găsită în [DANC82a] şi [DANC82b].

Programul DEBUGGER este o dezvoltare a monitorului prezentat în [DANC8o].

In cele ce urmează, în descrierea comenzilor vor fi utilizate majuscule pentru caracterele strict necesare pentru recunoaşterea comenzii, restul caracterelor din identificatorul comenzii putînd fi omise. Parantezele drepte vor fi utilizate pentru desemnarea elementelor opţionale.

6.3.3. MICRODOS - microsistem de operare pe disc flexibil

Microsistemul de operare realizat permite în principal, gestionarea, manipularea şi modificarea fişierelor sursă. Organizarea adoptată pentru discul flexibil este următoarea:

- pistele 0, 1 - MICRODOS şi EDITOR
- pistele 2 - 1o - rezervate pentru alte programe binare.
- pista 2o - pistă directoare.
- pistele 21 - 76 - fişiere sursă.

Fişierele au o lungime maximă de o pistă şi fiecare pistă este ocupată în întregime de un fişier. Fişierele sînt identificate după numele format din (1 ÷ 1o) caractere alfanumerice, primul fiind în mod obligatoriu un caracter alfabetic, majusculă.

Sistemul de operare acceptă următoarele comenzi din partea operatorului:

a. Read FILENAME

Are ca efect citirea unui fişier de pe disc în memorie.

In cazul în care fişierul cerut nu există pe disc se tipăreşte mesajul "FILE NOT FOUND" şi se aşteaptă o nouă comandă.

b. Write FILENAME

Are ca efect înscrierea textului sursă din memorie pe disc, cu numele FILENAME.

In scopul minimizării posibilităţilor de operare greşită, MICRODOS face următoarele verificări:

1 - existenţa unui text sursă în memorie, prin:

-lungime fişier diferită de zero,

-poziţie corectă a mărcii de sfîrşit de fişier.

In cazul în care sînt îndeplinite ambele condiţii de mai sus se tipăreşte mesajul "NO FILE IN MEMORY".

2 - inexistenţa pe disc a unui fişier cu acelaşi nume. In cazul în care pe disc există deja un fişier cu acest nume se cere permisiunea de supra-înscriere prin mesajul "DELETE OLD FILE ?".

3 - faptul că mai există loc pe disc. In cazul în care toate poziţiile de pe disc sînt ocupate se tipăreşte "DISK FULL" şi se revine în dialog fără alterarea textului sursă din memorie.

c. DELete FILENAME

Are ca efect ştergerea unui fişier de pe disc. In realitate fişierul nu este efectiv şters ci numele său este şters din pista directoare iar pista respectivă este marcată ca pistă liberă.

In cazul în care fişierul nu există pe disc se tipăreşte mesajul "FILE NOT FOUND".

d. Print FILE 1 [FILE 2] [FILE 3]...

Are ca efect tipărirea pe acel periferic asignat ca ieşire a fişierelor indicate.

e. INIT DISK

Are ca scop formarea şi iniţializarea unei noi dischete. Pentru a evita iniţierea operaţiei de mai sus şi distrugerea informaţiei de pe o dischetă printr-o manevră greşită, comanda este recunoscută numai dacă este introdusă exact în forma de mai sus (9 caractere).

Copierea celor 2o piste cu programe binare presupune 5 introduceri ale fiecărei dischete, datorită faptului că sistemul dispune de o singură unitate de disc flexibil.

Operatorul este ghidat prin mesajele:

INSERT NEW DISK.READY ?

INSERT OLD DISK.READY ?

La terminarea operaţiei se tipăreşte mesajul "O.K."

f. Cat

Pe acel periferic asignat ca ieşire se vor lista numele discului, numele tuturor fişierelor existente pe disc precum şi pistele pe care se află.

g. Output perif.1 [u perif.2]

Este comanda prin care poate fi asignat perifericul de icşire. Consola serie cu imprimantă (TTY), Display-ul, sau ambele simultan. In cazul în care are loc blocarea dialogului cu Display-ul paralel se face în mod automat dezafectarea lui şi asignarea consolei serie ca periferic de ieşire.

h. Debugger

Această comandă transferă controlul programului DEBUGGER rezident în memoria EPROM. Nu este folosită decît pentru punerea la punct a unor programe binare.

i. Editor

Această comandă transferă controlul programului EDITOR, de asemenea rezident în RAM. Revenirea din EDITOR se face tot în MICRODOS.

j. Interpreter

Are ca efect încărcarea de pe disc şi lansarea în execuţie a programului INTERPRETER, pentru execuţia programului de test.

6.3.4. Editorul de text

6.3.4.1. Descriere generală

Editorul a fost conceput şi realizat ca un editor de linie, în sensul că toate operaţiunile se fac la nivelul unei linii de text.

Organizarea memoriei de lucru este prezentată în fig. 6.5.

Orice linie introdusă în zona de lucru este încheiată de un caracter CARRIAGE RETURN (ODH), fără avans de linie, acesta adăugîndu-se automat la tipărirea sau afişarea liniei.

Sfîrşitul textului este marcat, în afara pointerului de sfîrşit de fişier (EOPTR) şi printr-un octet marcă de sfîr-

şit de fişier (0FFH). S-a ales această variantă, cu redondanţă, deoarece ea permite reducerea lungimii programului EDITOR.

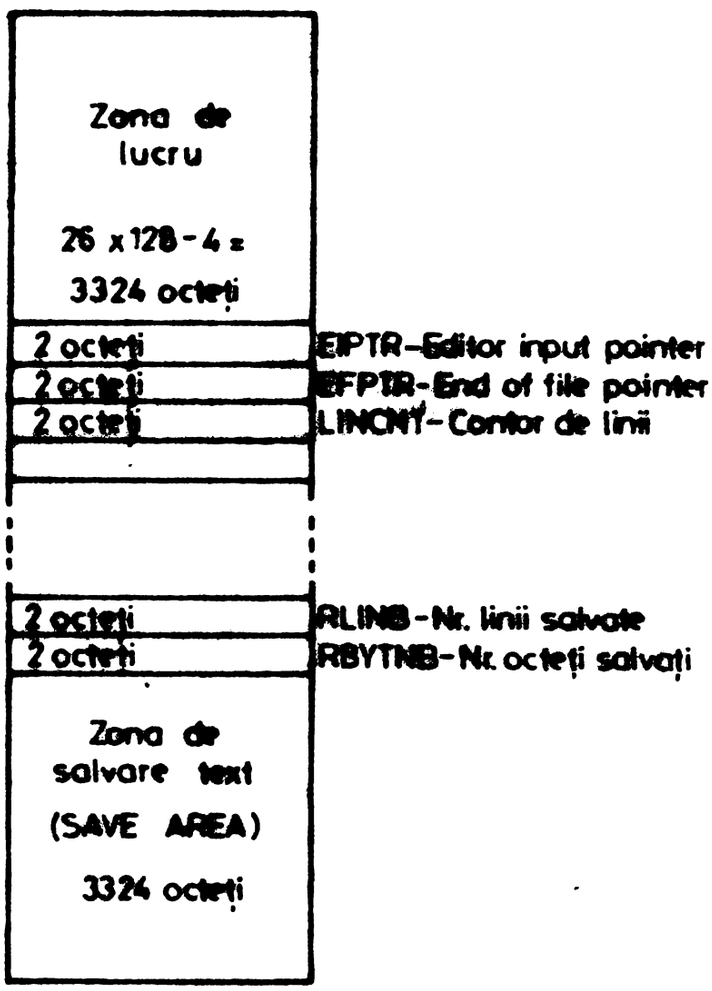


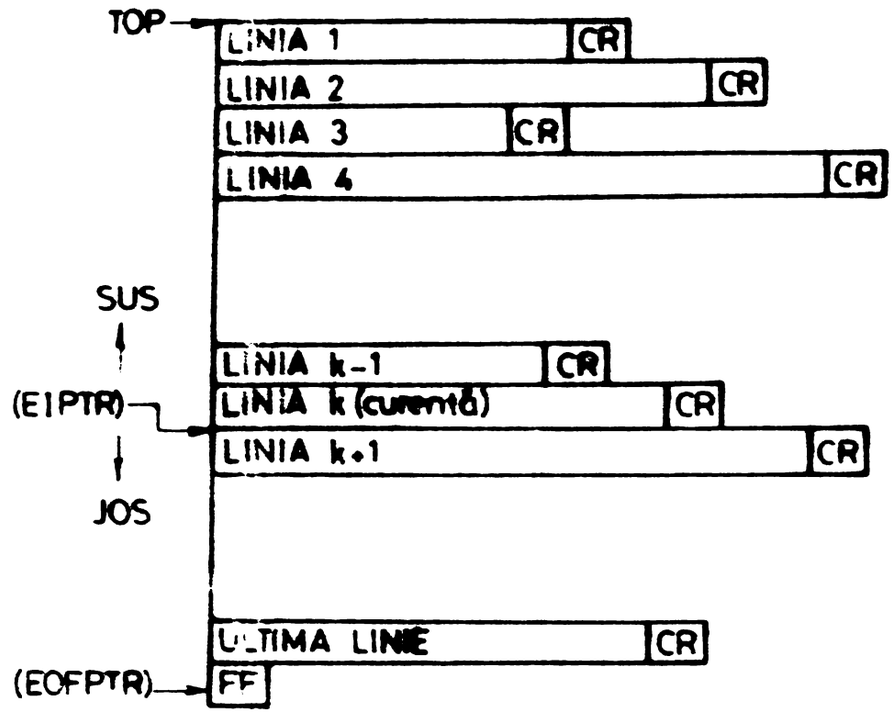
Fig.6.5. Alocarea memoriei de lucru

Variabila EIPTR, (Editor Input PoinTeR), marchează locul în care urmează să fie introdus (inserat) un text. Acest pointer indică adresa primului caracter din linia următoare liniei curente.

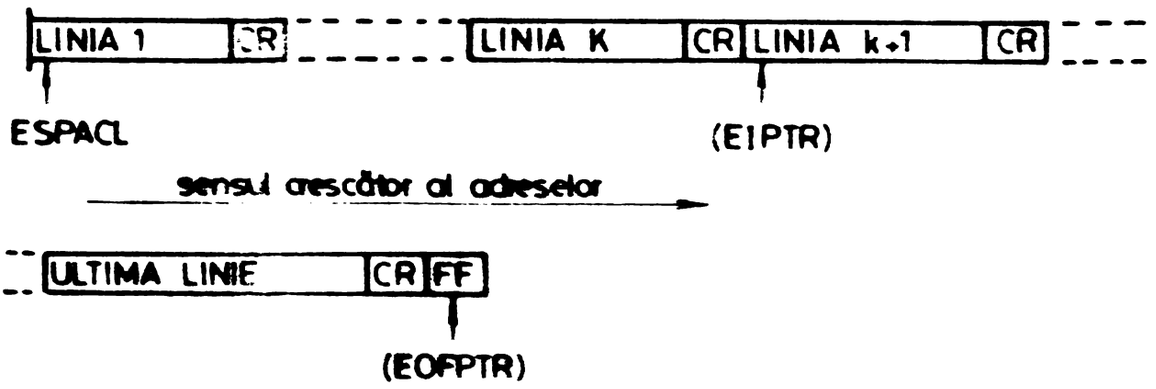
Variabila **LINCNT**, contor de linii, indică numărul liniei curente (cea aflată în faţa pointerului EIPTR).

Variabile EFPTR indică adresa mărcii de sfîrşit de fişier (fig.6.6).

Interfaţa operator a editorului este, ca şi în MICRODOS, subrutina GET, atît pentru comenzi cît şi pentru introducere de text.



a. Sensul direcțiilor SUS și JOS la deplasarea pointerului în text



b. Succesiunea liniilor în memorie și pozițiile pointerilor

Fig.6.6. Reprezentarea unui text în memorie

6.3.4.2. Comenzile editorului

Editorul acceptă următoarele comenzi:

a. Input sau Insert

După această comandă, editorul transmite mesajul INPUT şi trece în regimul de introducere/inserare text. Nu pot fi introduse/inserate fracţiuni de linie ci numai un număr de linii complete.

Ieşirea din regimul INPUT se face prin introducerea unei linii de lungime 0 (formată numai din caracterul CARRIAGE RETURN).

O linie este efectiv introdusă în spaţiul de editare, numai după ce a fost încheiată prin CR, pînă atunci ea putînd fi corectată prin metodele din § 6.3.5.

Pentru a fi înserată în text linia încheiată prin CR, întreg textul de după linia curentă (aflat în memorie între adresele (EIPTR) şi (EFPTR)) este deplasat în jos cu numărul de caractere ale noii linii, după cum noua linie este mutată din bufferul de intrare al subrutinei GET în spaţiul astfel creat. Apoi sînt ajustaţi pointerii EIPTR şi EFPTR şi contorul de linii şi se aşteaptă o nouă linie.

Inainte de începerea operaţiei de inserare se verifică dacă mai există suficient spaţiu în memoria de lucru. In caz că nu există spaţiu suficient se transmite "SPACE ERROR" şi se iese din regimul de INPUT. Astfel sînt protejate zonele de memorie învecinate şi nu este pierdută decît ultima linie introdusă.

b. DEL ete [nn]

Această comandă şterge din text numărul de linii indicat, începînd cu linia curentă. Indicarea numărului de linii este opţională, fiind presupus nn=1.

c. Top

Poziţionează pointerul deasupra primei linii de text.

d. Next [nn]

Poziţionează pointerul cu nn linii mai jos. In cazul în care prin această comandă se ajunge la sfîrşitul fişierului, pointerul este poziţionat după ultima linie din fişier

şi se tipăreşte, în afară de noua linie curentă, mesajul "EOF".

e. Up [nn]

Poziţionează pointerul de intrare cu nn linii mai sus. Dacă prin această comandă se atinge începutul fişierului se transmite mesajul "TOP".

f. Go [nn]

Are ca efect poziţionarea pointerului pe linia nn.

g. Output perif.1[,perif.2]

Comandă de asignare a perifericului de ieşire, similară celei din MICRODOS.

h. Find /ARTICOL/

Poziţionează pointerul pe linia care începe cu "ARTICOL" Se utilizează pentru căutarea unei linii din text. Căutarea începe din linia imediat următoare liniei curente.

i. Print [nn], [N]

Are ca efect tipărirea, pe acel periferic asignat ca ieşire, a nn linii din text, cu deplasarea pointerului pe ultima linie tipărită, care devine astfel linia curentă. In funcţie de optiunea [N], tipărirea se face cu sau fără număr de linie. Dacă în cursul operaţiunii se ajunge la sfîrşitul fişierului se tipăreşte mesajul "EOF".

j. Corection

In urma acestei comenzi editorul intră în regimul de corecţie (editare) a liniei curente. Cel mai simplu mod de corecţie este rescrierea întregii linii. De fapt, în acest regim se construieşte o nouă linie formată din:

- caractere introduse de la consolă,
- caractere din vechea linie, în cazul în care de la consolă se introduce caracterul "^".

Caracterele deja introduse în noua linie, (şi afişate pe ecran/tipărite la consolă), pot fi şterse unul cîte unul cu BACKSPACE (08H) sau RUBOUT (7FH). Noua variantă a liniei de text este construită în bufferul de intrare al subrutinei GET. Corecţia se încheie în momentul introducerii în noua linie a caracterului CR, fie de la consolă fie provenind din vechea linie.

Odată obţinută noua formă a liniei curente, editorul trece la ştergerea vechii linii din text şi inserarea celei noi ca şi în cazul comenzii INSERT.

De o tratare specială se bucură corecţia primei, penultimei şi ultimei linii din text deoarece ajustarea pointerilor şi a contorului de linii se face în mod diferit.

k. Put [nn]

Are ca efect deplasarea într-o zonă de salvare a unui număr de nn linii, începînd cu linia curentă, şi deplasarea pointerului pe ultima linie salvată.

l. GET

Are ca efect inserarea în text, după linia curentă, a liniei depuse în zona de salvare printr-o comandă PUT şi deplasarea pointerului pe ultima linie înserată. Comanda este importantă şi nu este recunoscută în cazul în care nu au fost depuse linii de text în zona de salvare, printr-o comandă PUT anterioară.

Aceste două comenzi sînt deosebit de utile deoarece ele permit:

- deplasarea unui bloc întreg de text dintr-o zonă în alta a unui fişier şi prin aceasta reorganizarea sa, fără rescrierea liniilor respective;
- deplasarea unui bloc întreg de text dintr-un fişier în altul. La comanda Quit, de revenire din editor în MICRODOS, zona de salvare nu este distrusă. La intrare în programul EDITOR, după citirea de pe disc a unui alt fişier, operatorului i se cere permisiunea de ştergere a zonei de salvare. El poate astfel introduce blocul de text salvat printr-o comandă PUT în noul fişier.

Toate comenzile, care au ca efect deplasarea pointerului de intrare, tipăresc noua linie curentă cu numărul ei.

m. Quit

Are ca efect transferarea controlului către programul apelant.

In continuare va fi prezentată subrutina principală de interfaţă operator.

6.3.5. Subrutina GET

Subrutina GET este calea principală de intrare, atît pentru introducerea comenzilor, cît şi pentru introducerea textului, în cazul editorului.

Această subrutină emite un caracter prompter, apoi acceptă caractere de la consolă pe care le depune succesiv într-o zonă de memorie, INPUT BUFFER, pînă la introducerea caracterului CR. Se execută apoi revenirea în programul apelant, avînd în registrul acumulator primul caracter introdus (pentru identificarea rapidă a unei comenzi) iar în registrul B, lungimea liniei introduse.

O tratare specială o au caracterele BACKSPACE (08H), RUBOUT (7FH) şi semnul exclamării (21H).

Caracterul BS este transmis ca atare la perifericul asignat ca ieşire şi are ca efect decrementarea contorului de caractere şi a pointerului de intrare INPUT POINTER (ştergerea ultimului dintre caracterele deja introduse în linie). Acesta este un mod comod de a corecta o linie în cursul introducerii ei, atunci cînd perifericul de ieşire este un display.

Corecţia liniei în acest mod ar fi dificilă atunci cînd se lucrează cu imprimanta, deoarece, pe de o parte aceasta nu are un cursor care să indice locul unde urmează să fie tipărit următorul caracter, iar pe de altă parte prin deplasarea înapoi a capului de imprimare caracterele nou introduse ar fi tipărite peste cele vechi, făcînd greoaie aprecierea conţinutului liniei.

Din motivul de mai sus, s-a adoptat încă un caracter cu aceeaşi funcţie ca şi backspace (rubout=7FH), care diferă de primul doar prin modul de afişare a corecţiei; la perifericul de ieşire se transmite caracterul şters, prin aceasta evitîndu-se supra-imprimarea. Astfel, pe imprimantă, o linie corectă are următorul aspect:

```
tastatură      - ACEASTA ESTE OC_ LIMIE__NIE
tipărire text  - ACEASTA ESTE OCC LIMIEEIMNIE
efectiv                                    _ = rubout
introdus       - ACEASTA ESTE O LINIE
```

Caracterul "!" are ca efect anularea întregii linii in-

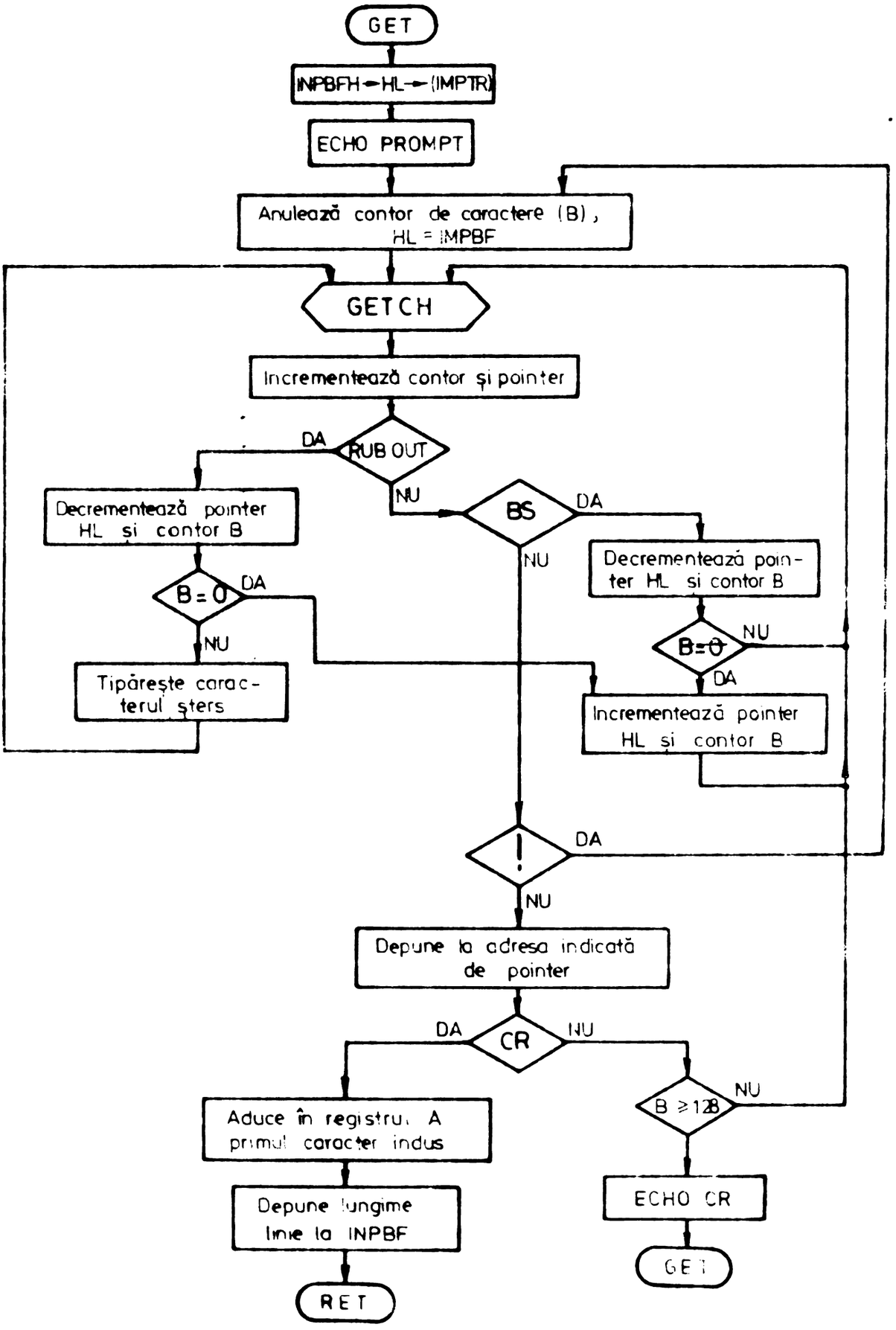


Fig. 6.7. Subrutina GET

trecuse pînă în acel moment, prin reiniţializarea pointerului şi a contorului de caractere (caracter "line delete").

Organigrama subrutinei este prezentată în fig.6.7.

6.3.6. Consideraţii generale asupra sistemului de operare şi editorului

Sistemul de operare şi editorul sînt rezidente permanent în RAM şi ocupă un spaţiu de numai 3,5 KO. Concentrarea într-un spaţiu atît de restrîns a fost posibilă prin:

a. - Alcătuirea tuturor comenzilor ca subrutine, astfel încît să poată fi apelate din orice punct al programului. De exemplu comenzilor Output din MICRODOS, EDITOR şi TEST INTERPRETER le corespunde aceeaşi subrutină.

b. - Realizarea modulară a programului, bazat pe subrutine generale, cu grad crescînd de complexitate şi puncte de intrare multiple.

c. - Existenţa unei singure rutine de tratare a erorilor, comună tuturor programelor. Acestei rutine i se modifică dinamic valorile cu care trebuie reiniţializaţi anumiţi parametrii, ca de pildă stiva şi punctul de revenire.

Editorul oferă maximum de flexibilitate şi simplitate în operare, fiind (prin structura subrutinei de interfaţă operator şi prin comenzile PUT şi GET) superior editorului cu care este livrat sistemul de operare ISIS-II (INTEL) respectiv SFDX-18 (microcalculatorul M18-FCE) [INTE76], [FCE78]. Din aceste puncte de vedere, el este comparabil cu editoarele ce însoţesc sistemele de operare pentru procesoare din clasele superioare, cum ar fi MCS.1 0 (Zilog), sau RSX-11 (DEC) [DEC77], [ZILO77].

Cu deosebită grijă au fost tratate toate cazurile de operare eronată, în aşa fel încît să se evite pierderea textului introdus în memorie sau distrugerea unui fişier. Operatorul este informat întotdeauna asupra tipului de eroare comisă, astfel încît el poate repeta comanda dorită.

S-a avut în vedere acest lucru pentru a face posibilă exploatarea familiei de testoare numai de către personalul cu calificare medie.

Sistemul de operare, în ciuda simplităţii sale, satisface complet necesităţile familiei de testoare realizate de

I.T.C., filiala Timişoara.

6.4. Structura interpretorului

6.4.1. Descriere generală

Ca şi sistemul de operare MICRODOS, interpretorul pentru limbajul de test MTL are o structură modulară, reprezentată în fig.6.8.

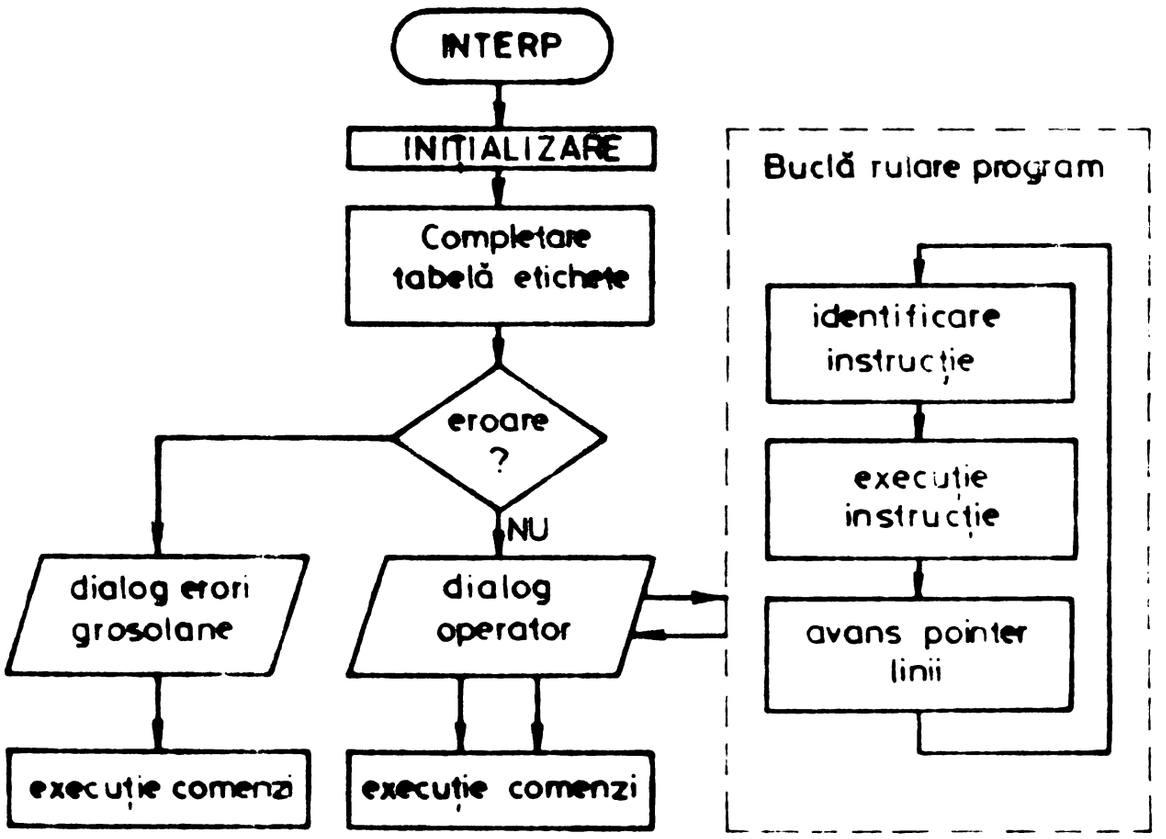


Fig. 6.8. Structura interpretorului

După iniţializare se face o primă analiză a programului şi se completează tabela de etichete. In cazul unei erori grosolane (etichetă multidefinită, tabelă plină), se trece în regimul de dialog special destinat acestui scop. In acest regim

sînt acceptate numai acele comenzi care pot duce la modificarea programului existent în memorie, sau revenire în MICRODOS (citire program de pe disc şi editare).

Dacă la completarea tabelei de etichete nu s-au detectat erori, se intră în regimul de dialog operator, în care se acceptă comenzi.

In urma unora dintre comenzi, interpretorul intră în bucla de reluare program, care cuprinde interpretarea şi execuţia programului, linie cu linie, în mai multe etape:

- filtrarea tuturor caracterelor din linie pînă la începutul zonei instrucţie
- identificarea instrucţiei
- apel la subrutina specifică de execuţie a instrucţiei, aceasta interpretează zona operand şi execută propriu-zis instrucţia
- avansul pointerului de linii pe linia următoare şi reluarea buclei.

Interpretorul acceptă, aşa cum a fost expus în capitolul 6.2.1 următoarele întreruperi:

STØP - oprire rulare program şi revenire în regimul de dialog

START - lansare în execuţie a programului de test începînd cu linia 1. Echivalentă cu comenda RUN

RELUARE - relansare în execuţie a programului de la linia curentă, după o oprire programată. Echivalentă cu comanda GO

ERH - Eroare hard. Această întrerupere este furnizată de baza de timp, (de către un dispozitiv tip watch-dog), sau de către cuplor, pentru a semnaliza regimuri anormale ale acestora.

TT - terminare secvenţă test a procesorului rapid.

TEMP - cerere de temporizare din partea procesorului rapid. Temporizările lungi sînt realizate de către microprocesor, fiind programate în durată cu instrucţia FPW

Din punct de vedere al întreruperilor sistemul se poate afla în una din următoarele 4 stări:

STAREA 0

Nu este acceptată nici una din întreruperile de mai sus. Această stare este inițializată de către MICRODOS și este menținută pînă la intrarea în regimul normal de dialog (completare tabelă etichete, inițializări variabile etc.) și este refăcută de comenzile care duc la părăsirea (chiar și temporară) a interpretorului.

STAREA 1

Intreruperi acceptate: STOP, START, REL, ERH. Interpretorul se află în această stare în regimul de dialog operator.

STAREA 2

Intreruperi acceptate : STOP, ERH, TT. TEMP. Interpretorul se află în această stare numai pe durata execuției unui test de către procesorul rapid, adică o parte din timpul afectat instrucției COD.

STAREA 3

Intreruperi acceptate : STOP, ERH. Starea 3 apare pe durata rulării , mai puțin execuția instrucției COD și regimul de dialog.

Cele 4 stări sînt rezumate în tabelul 6.1.

Tabelul 6.1. Stările regimului de întreruperi.

Intr. / Stare	TEMP	TT	ERH	REL	START	STOP	Masca
INTSTO	NU	NU	NU	NU	NU	NU	1111 1111
INTST1	NU	NU	DA	DA	DA	DA	1111 0000
INTST2	DA	DA	DA	NU	NU	DA	1100 0110
INTST3	NU	NU	DA	NU	NU	DA	1111 0110

6.4.2. Comenzile interpretorului

In cazul în care, în cursul completării tabelei de etichete sau a reluării programului, s-au detectat erori în program, se intră într-un regim de dialog special destinat acestui caz. Sînt acceptate următoarele comenzi:

a. Quit - părăsirea interpretorului şi transferul controlului către MICRODOS.

b. Editor - transferul temporar al controlului către EDITOR pentru modificarea programului aflat în memorie. In urma comenzii Quit din EDITOR, revenirea se face tot în INTERPRETER.

c. Input FILENAME

Citirea de pe disc a unui program.

Comenzile de mai sus sînt acceptate şi în regimul normal de dialog.

Operatorul este avertizat asupra intrării în regimul de dialog pentru erori prin:

- tipărirea timpului de eroare şi a liniei de program care o conţine,
- tipărirea mesajului "MUST ENTER A FILE",
- schimbarea caracterului "Prompter",

In regimul normal de dialog operator sînt acceptate comenzile de mai sus, precum şi:

d. Run

Are ca efect iniţializarea procesorului rapid, a stivei interpretorului şi a tuturor variabilelor, urmată de lansarea în execuţie a programului, începînd cu linia 1.

e. Go - lansarea în execuţie a programului începînd cu linia pe care a fost oprit.

f. Step - execuţia unui pas (linie) de program.

g. Output perif.1 [perif.2] - asignare periferic ieşire: comandă identică cu cea din MICRODOS.

h. Print - tipărirea liniei ce urmează a fi executată.

i. Print M - validarea sau invalidarea execuţiei instrucţiei NAME (M = All sau M = Nothing).

j. SEria X.. - introducerea seriei plăcii testate pentru a fi tipărită la execuţia instrucţiei TITLE.

k. Bem - tipărirea/afisarea hărţii de erori a plăcii.

Intreruperile START şi RELUARE au rolul de comenzi, fiind echivalente cu RUN respectiv GO. Intreruperea STOP opreşte rularea programului şi decuplează sursele de alimentare ale plăcii.

6.4.3. Structura buclei de rulare program

Organigrama simplificată a acestei bucle - nucleul de bază al interpretorului - este prezentată în fig.6.9.

Ea prezintă în principal trei puncte de intrare, corespunzătoare comenzilor RUN, GO şi STEP.

Ca urmare a comenzii RUN se execută întîi o secvenţă de iniţializare a procesorului rapid urmată de iniţializarea unor variabile ale sistemului:

- stiva instrucţiilor CALL.
- pointerul de linii.
- fanion pentru execuţia pas cu pas.
- bufferele de text ale instrucţiilor TITLE şi NAME.

După cîteva operaţii se trece la interpretarea liniei curente. Dacă aceasta nu conţine cîmp instrucţie se deplasează pointerul pe linia următoare şi se reia această buclă internă.

In cazul în care există cîmp instrucţie se apelează subrutina de identificare a instrucţiei.

Dacă rezultatul este pozitiv, după verificarea fanionului SSTFLG se trece la execuţia propriu-zisă a instrucţiei.

Fiecărei instrucţii îi corespunde o subrutină a cărei adresă este furnizată de subrutina de identificare a instrucţiei. Se avansează pointerul de linie şi se revine în punctul INTO1.

In cazul în care fanionul SSTFLG a fost 1 s-a tipărit linia respectivă înainte de execuţia ei.

Se face o nouă verificare a acestui fanion. Dacă el este "1" (pas cu pas) se iese din buclă, se anulează fanionul, deoarece pasul a fost executat şi se revine la regimul de dialog operator.

Similară este parcurgerea buclei pentru cazul comenzii STEP care poziţionează pe "1" fanionul SSTFLG sau GO, care permite reluarea rulării programului începînd cu linia următoare, fără iniţializarea variabilelor.

In această buclă principală există două posibilităţi de ieşire cu eroare

- întîlnirea mărcii de sfîrşit de fişier
- nerecunoaşterea setului de caractere din cîmpul instrucţiune (instrucţie necunoscută).

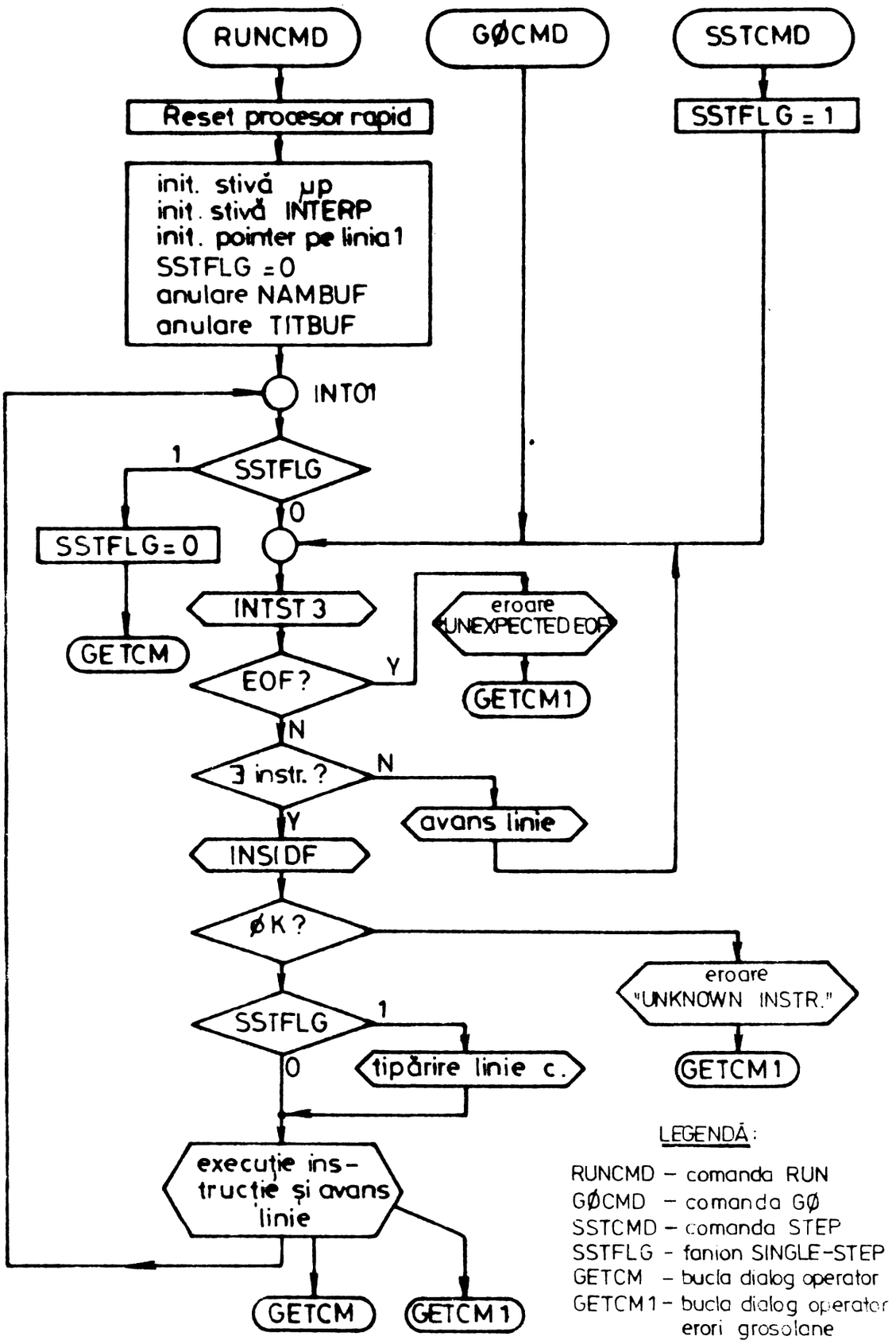


Fig.6.9. Organigrama simplificată a buclei de rulare program

De asemenea, fiecare subrutină de execuţie a instrucţiei face verificarea sintactică a cîmpului operand, transferînd controlul unei proceduri unice, în cazul sesizării unei erori.

Toate aceste erori, cuprinse în programul sursă, determină reiniţializarea procesorului rapid şi a surselor programabile şi apoi intrarea în regimul de dialog special, amintit anterior.

Ieşirea normală din buclă, cu afişarea rezultatului testării, este determinată de instrucţia END.

In afară de aceasta, există instrucţiuni care produc ieşirea temporară din buclă (oprirea rulării) şi intrarea în dialog, de pildă STP, OPI şi, în anumite condiţii, instrucţiunea COD.

6.4.5. Prezentarea rezultatelor testării

Rezultatele testării pot fi prezentate pe durata rulării programului, în mai multe moduri, funcţie de poziţia unor chei de pe panoul de comandă.

a. La terminarea fiecărui test (cod), în cursul căreia au fost detectate erori, se tipăreşte o linie care conţine

- parametrii cpdului sau ai erorii, după caz, de pildă :valoarea constantei de temporizare, dacă s-au executat temporizări, sau rangurile liniilor de adrese defecte la testele AFP,
- localizarea erorii în jumătatea inferioară sau superioară a cuvîntului de date,
- numele codului,
- valorile tensiunilor de alimentare.

Această tipărire poate fi inhibată prin poziţionarea unei chei de pe panou.

b. La terminarea fiecărui test (cod), în cursul căruia au fost detectate erori, se tipăreşte harta cu C.I. de memorie defecte (BOARD ERROR MAP). Această tipărire este condiţionată de poziţia unei chei de pe panoul testorului.

c. In cursul execuţiei instrucţiei BEM se tipăreşte/afişează harta cumulativă a C.I. defecte, condiţionată de poziţia cheii CBM. de pe panou.

In toate cazurile în care tipărirea BEM se face numai pe display, se opreşte bucla de rulare pentru interpretarea rezultatelor afişate. Relansarea se face de către operator, prin apăsarea tastei RELUARE. In cazul în care tipărirea se face numai, sau şi, pe imprimantă rularea nu este oprită, sistemul putînd funcţiona nesupravegheat.

d. La execuţia instrucţiei END, rezultatul final testării (PASS/FAIL) este tipărit întotdeauna şi pe imprimantă, indiferent care periferic este asignat ca periferic de ieşire.

Datorită facilităţilor de mai sus, sistemul de test duce la o productivitate foarte ridicată, prin utilizarea în următorul mod:

a. In prima etapă, de punere la punct şi detectare a defectelor ferme, prin algoritmi de test de durată mică, este necesară afişarea foarte frecventă a BEM, practic după fiecare test. Se lucrează cu display-ul ca periferic de ieşire, deoarece afişarea unui BEM durează cca. o,1 s, comparabil cu durata testelor.

b. După remedierea acestor defecte ferme, se trece la execuţia testelor lungi, inhibînd tipărirea BEM după fiecare test şi asignînd şi imprimanta ca periferic de ieşire. Tipărirea unui BEM durează acum cca. 15 secunde (4 rînduri a 8o caractere), mult mai puţin decît durata testelor cu secvenţă de adresare de tip GALLOPING [DANC83a].

Această etapă durează cca. 3o minute pentru unităţi de memorie de 16 K cuvinte. Testorul poate lucra nesupravegheat, operatorul putînd părăsi locul de la consolă, în vederea remedierii defectelor constatate pe alte unităţi de memorie.

6.4.6. Implementarea instructiei COD

Această instrucţiune este cea mai complexă, din punct de vedere al execuţiei, dintre instrucţiunile limbajului MTL, fiind în fapt o macroinstrucţiune. Se reaminteşte sintaxa ei (4.48):

<instr. exec. secvenţă> ::=
COD <sep> <nume cod> [<sep> I]

Execuţia ei se face în mai multe etape, o enumerare simplificată fiind cea de mai jos:

a. Iniţializarea zonei de microprogram a procesorului rapid.

b. Se verifică dacă cheile speciale pentru lansarea de la panou a anumitor coduri nu sînt active.

c. Se identifică <nume cod> în tabela de coduri.

d. Se încarcă microcodul (microprogramul) corespunzător în memoria de program a procesorului rapid.

e. Se lansează în execuţie codul şi se aşteaptă întreruperea de terminare a secvenţei (TT) din partea procesorului rapid.

f. Se interpretează rezultatul şi se tipăreşte eroarea, dacă este cazul.

g. Dacă întreruperea a fost de terminare parţială, atunci, după parcurgerea subrutinei de întrerupere specifică sau nespecifică, se încarcă următoarea secvenţă şi se reia de la punctul e.

Deoarece instrucţiunea COD a fost implementată prin cca. 600 linii program la care se adaugă cca. 300 linii pentru subrutinele de încărcare microcod şi afişare BEM, o prezentare detaliată nu-şi are locul aici. Se va face doar o prezentare voit simplificată. Organigrama simplificată este prezentată în fig.6.10.

Pe durata execuţiei codului de către procesorul rapid, microprocesorul aşteaptă în starea HALT întreruperea şi terminarea secvenţei (TT). A fost aleasă această modalitate deoarece permite servirea cea mai rapidă a cererii de întrerupere. Un test poate fi compus din una sau mai multe secvenţe de test, care se încarcă şi se execută succesiv, de către procesorul rapid. Intreruperea poate fi deci de două feluri, terminare parţială, sau terminare test. Acest lucru este marcat de poziţia unui bit, dintr-un registru de stare al procesorului rapid.

Pentru anumite coduri, cum ar fi de pildă testul liniilor de adresă [DANC03a], [CHAS77a], [CHAS77b], este necesară interpretarea rezultatului, după terminarea fiecărei secvenţe. Sînt necesare deci subrutine specifice fiecărui cod, în cazul în care acesta necesită o anume interpretare a rezultatelor

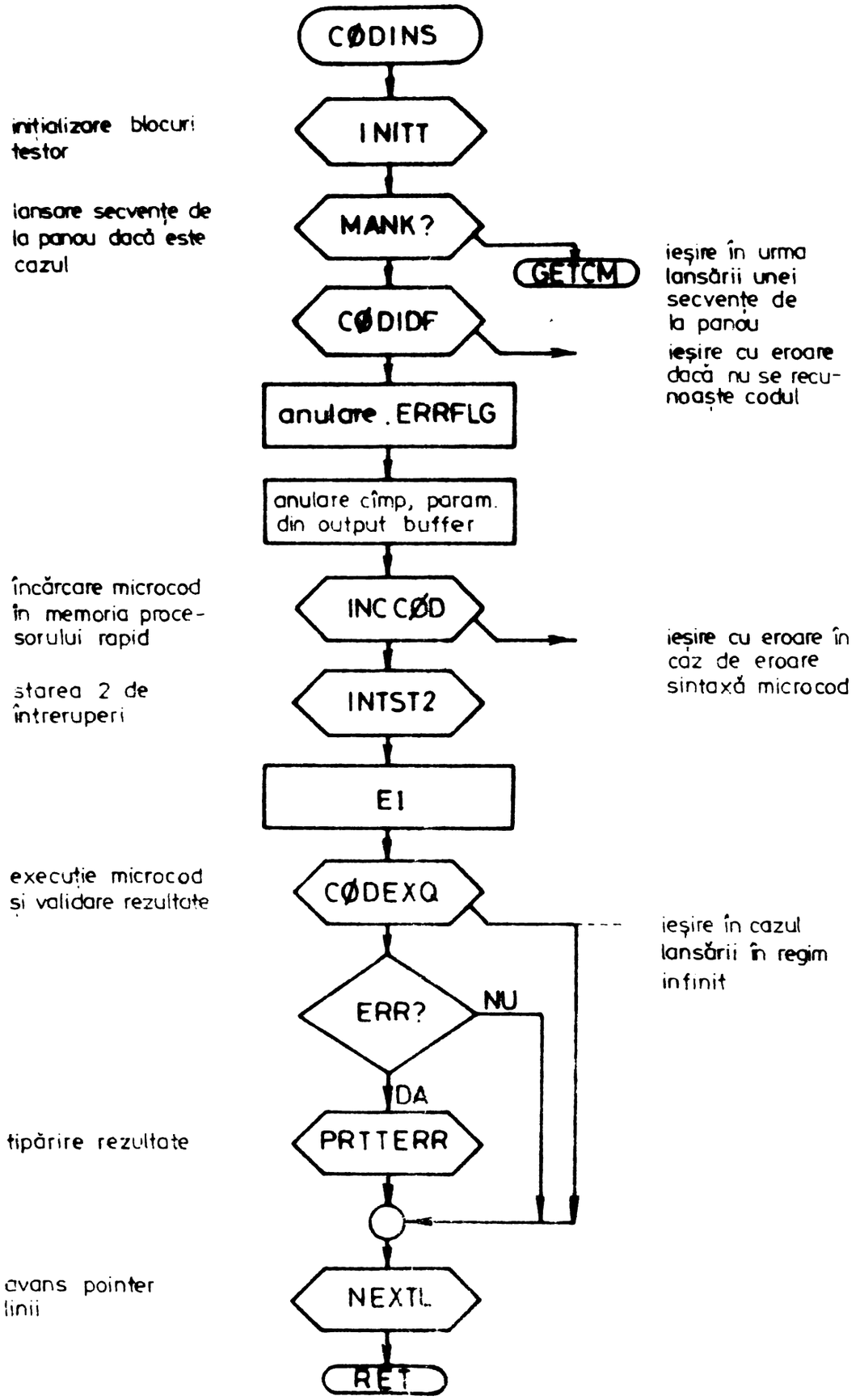


Fig. 6.10. Organigrama simplificată a subrutinei corespunzătoare instrucției CØD

testării, după fiecare secvenţă. Aceste subrutine vor fi denumite în continuare subrutine specifice. In cazul în care nu este necesară interpretarea rezultatelor după fiecare secvenţă, tratarea întreruperii este transferată unei subrutine nespecifice, care încarcă următoarea secvenţă de microprogram şi o lansează în execuţie.

Există teste, pentru verificarea unor anumiţi parametrii ai selecţiei unităţilor de memorie, care trebuie să ducă la erori dacă unitatea testată este lipsită de defecte. Aceste coduri se vor numi <u>inverse</u> şi se va ţine seama de acest lucru în interpretarea rezultatelor.

Modul de reprezentare a parametrilor microcodului în tabela de coduri este original şi asigură identificarea tipului de subrutină specifică/nespecifică, precum şi a tipului de cod, normal/invers, printr-o tratare simplă şi unitară.

```
DB 'NUMCOD' ; Numele codului (test pattern)
DW ADRCOD   ; Adresa din memorie unde se află micro-
            ; codul
DB TIP      ; Tipul algoritmului (normal/invers)
DW SSPEC    ; Adresa subrutinei specifice. Dacă
            ; SSPEC = 0000 se implică subrutina
            ; nespecifică
```

In cazul în care în momentul execuţiei instrucţiei COD este activă una din cheile pentru lansarea de la panou a unor teste speciale, (de stimulare în vederea depanării), se încarcă şi se lansează codul cerut de la panou în regim infinit (cu autobuclarea procesorului rapid). Operatorul este anunţat de acest lucru prin mesajul "START OF KEYBOARD TEST # n". In acest moment, cheile care condiţionau tipărirea rezultatelor au semnificaţii diferite:

- anularea strobării adreselor
- anularea strobării datelor de intrare
- suprimarea regenerării comandate de testor
- suprimarea modului dialog (hand-shake), în cazul în care se testează module de memorie cu hand-shake, ce emit semnale tip BUSY şi END OF CYCLE.

In acest caz nu se face avansul pointerului de linii, putîndu-se relansa alt cod în aceleaşi condiţii de timing şi tensiuni de alimentare, prin apăsarea tastei RELUARE.

6.5. Concluzii

Sistemul software, conceput de autor pentru testoarele de memorii din familia MOSTEST-o3, este alcătuit din trei module :

- un sistem de operare pe disc flexibil,
- un editor de text interactiv, superior din multe puncte de vedere editoarelor uzuale implementate pe microprocesoare de o biţi,
- un interpretor de test - executiv pentru limbajul de test MTL.

Programul sursă în limbaj de asamblare cuprinde cca. 69oo linii din care peste 6ooo aparţin autorului. Prin combinarea tehnicilor de proiectare sus-jos şi jos-sus ("top-down" şi "down-top") care a dus la subrutine cu puncte de intrare multiple, apelabile din orice punct al programului, codul obiect propriu-zis a putut fi condensat într-un spaţiu de numai 8.5 KO, la care se adaugă tabelele şi variabilele sistemului precum şi microcodurile, ajungînd la un total de 16 KO. Sistemul necesită un minim de 24 KO RAM, pentru a putea fi rulat.

Proiectarea sistemului software a avut în vedere, în afara cerinţelor de testare, în primul rînd orientarea către utilizator şi posibilitatea ca echipamentul să poată fi operat de personal cu calificare medie, adică acea sumă de caracteristici întrunite sub calificativul, asimilat ca atare, de "user friendly".

Autorul a adus mai multe contribuţii originale, în ceea ce priveşte arhitectura sistemului software, metodele de implementare, dintre care unele au fost prezentate sumar în acest capitol. Ele se referă, în special la modul de implementare a corecţiilor în cadrul editorului, transformarea BCD - binar şi binar - BCD (subrutinele CZHEX şi CHEXZ care nu au fost prezentate în acest capitol), implementarea opţiunii pas cu pas în bucla de rulare program, implementarea instrucţiunii CoD şi tratarea unitară a codurilor şi a întreruperilor. Este de asemenea, originală metoda de lansare a unor teste de la panoul testorului, cu modificarea condiţiilor de stimulare a memoriei testate.

In anexa A5.e este prezentat un exemplu concret de rulare a unui program de test pe echipamentul MOSTEST-o3D.

CAPITOLUL 7.

C O N C L U Z I I

Incepînd din anul 1970, anul fabricaţiei primului calculator în care memoriile cu C.I. au înlocuit feritele, s-a constatat că problemele testării UM cu C.I. sînt esenţial diferite de cele ale UM cu ferite, în primul rînd datorită imposibilităţii accesului la celula de memorare.

Au fost dezvoltate metode, atît pentru testarea 'GO/NO GO' cît şi pentru diagnoza UM, în scopul identificării şi remedierii defectelor.

Aşa cum a demonstrat J.P.Hayes în anul 1975, o testare completă nu este posibilă, datorită numărului mare de stări pe care le poate avea o memorie. Este posibil, şi probabil, ca un număr apreciabil de defecte, din clasa defectelor de sensibilitate la informaţia înscrisă (PSF), care au un timp de latenţă mare, să nu fie detectate în cursul testării.

Testarea trebuie să aibă ca prim scop eliminarea defectelor "certe" (cu timp de latenţă mic) şi apoi eliminarea acelor defecte cu timp de latenţă mare ("soft errors") care au probabilitate de apariţie mai redusă . Pentru ca acestea să fie cunoscute este necesară caracterizarea periodică a C.I. de memorie.

Pentru asigurarea unei rate cît mai mici a erorilor, producătorul are de ales între două variante: -o testare intensivă a UM, cu eliminarea tuturor C.I. de memorie care prezintă defecte şi-aplicarea unor metode de corecţie a erorilor. Aceasta din urmă poate să reducă rata erorilor cu cîteva ordine de mărime. Deşi a doua metodă presupune existenţa unui număr sporit de C.I. în produsul finit ea poate conduce la consumuri specifice mai mici. Decizia de adoptare a uneia sau al-

teia dintre metode este de obicei de ordin economic.

In afară de problemele testării GO/NO GO a UM, lucrarea de faţă prezintă unele aspecte ale testării cu diagnoză a UM, în vederea identificării şi remedierii defectelor.

Ca şi în celelalte ramuri ale producţiei de sisteme şi subansamble electronice, introducerea echipamentelor de testare automată duce la ridicarea productivităţii muncii, dar numai cu condiţia implementării lor corecte în fluxul de producţie. Pentru a obţine efectele economice dorite, introducerea ETA trebuie să fie însoţită de schimbări în structura personalului şi calificarea de deservire şi întreţinere. Aceste aspecte au fost prezentate în [LUPU82] şi reluate de autor în [DANC83g].

*

* *

Principalele contribuţii ale autorului se referă la :

- prezentarea pentru prima oară în literatura noastră a funcţionării C.I. de memorie RAM, statice şi dinamice (anexa 1) şi a defectelor acestora (capitolul 2 şi anexa 1);

- analiza comparativă a performanţelor testelor utilizate pentru memorii RAM semiconductoare, pe baza datelor publicate în literatură (anexa 2);

- generalizarea algoritmului Srini pentru diagnoza de adresă din UM, în aşa fel încît să poată detecta şi defecte de scurtcircuit între linii, atît pentru cazul dominanţei de 0 (funcţie logică SI) cît şi pentru cazul dominanţei de 1 (funcţie logică SAU). (§ 4.5.2, 4.5.3);

- elaborarea unui algoritm original - SC - pentru diagnoza liniilor de adresă din UM, care permite identificarea defectelor de scurtcircuit între ele, separat de defectele de blocare, atît pentru cazul dominanţei de 1 cît şi de 0 (§ 4.6) Algoritmul a fost verificat atît prin simulare (§ 4.6.4 şi anexele A4.a, A4.b) cît şi prin implementarea sa efectivă pe calculatorul I 100 (§ 4.6.6). Au fost definite cerinţele pe

care trebuie să le îndeplinească ETA pentru implementarea algoritmului SC (§4.6.5) şi limitele de aplicabilitate, datorate deplasării nivelelor de "1" şi "0" în cazul scurtcircuitului între un număr mare de linii (§ 4.7);

- a fost elaborat un aparat matematic original pentru descrierea defectelor de blocare şi de scurtcircuit pe magistrale (§ 4.5);

- acest aparat matematic a fost utilizat la demonstrarea a 15 teoreme originale referitoare la algoritmul Srini generalizat (T1 - T9, din § 4.5.2, § 4.5.3) şi la algoritmul SC elaborat de autor (T10 - T15, din § 4.6). Teoremele T1 şi T9 sînt valabile şi pentru alţi algoritmi de acest tip. A fost demonstrată pentru prima oară unicitatea mulţimilor de adrese pe care se poate desfăşura acest tip de teste. Pînă în prezent alegerea acestor mulţimi de adrese se făcea intuitiv;

- au fost definite necesităţile software ale unui echipament de testare automată a memoriilor (§ 5.1, § 6.3.1);

- a fost elaborat un nou limbaj de nivel înalt, MTL, pentru testarea UM, orientat spre utilizator (capitolul 5); Acest limbaj, implementabil pe microprocesoare de 8 biţi din familia 8080/8085 conţine toate instrucţiunile necesare oricărui tip de stimulare/evaluare a unei unităţi de memorie. Prin preluarea de către executiv a unor funcţii de stimulare/evaluare, numărul de instrucţiuni a putut fi redus, permiţînd astfel reducerea lungimii programelor de test şi a efortului de programare;

- a fost elaborat un sistem de software de bază, alcătuit din: sistem de operare pe disc flexibil, editor interactiv (§ 6.3) şi executiv pentru limbajul MTL (§ 6.4). Programul sursă în limbaj de asamblare este prezentat în anexa A6. După încărcarea programului de pe disc, operarea sistemului de test se poate face în totalitate de la un panou de comandă, care conţine numai 8 chei şi trei taste. Cu ajutorul acestora din urmă se comandă lansarea în execuţie şi oprirea programului; nu mai este necesară nici o intervenţie la consolă sistemului, ceea ce face posibilă operarea de către personal cu calificare medie sau joasă. Cu ajutorul cheilor de pe panou, operatorul poate influenţa desfăşurarea programului, sau poate lansa un set particular de secvenţe de stimulare pentru

depanare. Prin tratarea minuţioasă a tuturor erorilor posibile de operare şi programare, acestea nu duc la distrugerea programului de test sau a rezultatelor parţiale ale testării. Prin utilizarea unor metode originale de implementare, spaţiul de memorie ocupat a fost restrîns la 16.5 KO (din care 8 KO microcodurile - programe ale procesorului rapid de test) iar spaţiul necesar rulării la cca.24 KO.

Sistemul software, conceput şi realizat de autor, a fost implementat pe o familie de echipamente de testare automată a unităţilor de memorie. Aceste echipamente au fost realizate la I.T.C. filiala Timişoara. Dintre acestea, echipamentul MOSTEST-03D se află în exploatare la Fabrica de Memorii Electronice şi Componente pentru Tehnica de Calcul, cu varianta sistemului software prezentată în această lucrare, de peste un an, cu rezultate deosebite în ceea ce priveşte siguranţa în funcţionare flexibilitatea şi eficienţa testării.

8. LISTA PRESCURTARILOR UTILIZATE

Ca orice domeniu tehnic, şi în domeniul testării automate, viteza cu care pătrund noţiuni noi este mai mare decît cea cu care limba le poate traduce şi asimila. In faţa autorului au stat două alternative crearea de prescurtări noi, negeneralizate şi utilizarea celor din limbile de provenienţă a noţiunilor, utilizate deja pe larg de specialişti, atunci cînd o traducere a termenilor este încă ambiguă şi negeneralizată. Pentru uşurinţa lecturii, autorul a ales varianta utilizării traducerii în limba romană a noţiunilor concomitent cu utilizarea prescurtărilor utilizate în publicaţiile de circulaţie internaţională.

AFP - diagrama adreselor defecte (Address fault plot)

ATE - (Automatic Test Equipment) Echipament(e) de testare automată

BEM - (Board Error Map) Harta erorilor pe modul (de memorie)

CBB - circuit basculant bistabil

CE - (Chip Enable) - idem CS

CI - circuit integrat

CPU - (Central Processing Unit) Unitate centrală

CS - (Chip Select) Denumire a semnalului de selecţie a CI de memorie, de obicei activ pe 0

DUT - (Device under test) dispozitiv supus testării

ETA - Echipamente de testare automată

FDC - (Floppy disk controller) - interfaţa la unitatea de disc flexibil

I/O - Intrare/Ieşire

MTL - (Memory Test Language) Denumire a limbajului de testare a UM

MUT - (Memory Under Test) Memoria supusă testării

PIO - (Peripheral Input Output) Interfaţă programabilă de intrare/ieşire

PSF - (Pattern Sensitive Fault) defect de sensibilitate la informaţia înscrisă

RAM - (Random Access Memory) Memorie cu acces aleator

RD - denumire a semnalului de CITIRE (de obicei activ pe zero)

UC - unitate centrală

UM - unitate de memorie

UUT - (Unit under test) dispozitiv, ansamblu sau sistem supus testării

WR - denumire a semnalului de INSCRIERE (de obicei activ pe zero)

9. BIBLIOGRAFIE

1. [ABB081] - R.Abbot et.al., "Equiping a line of memories with spare cells", Electronics, Vol.54, No.15, 28 July 1981, pp.127-13o.
2. [ADAR8O] - DR 12/3o Memory test System Description, Adar-Scientific Atlanta, Burlington, MA, 198o.
3. [AHO77] - A.V.Aho & J.D.Ullmann, Principles of Compiler Design, Addisson - Wesley, Reading, MA, 1977.
4. [BALM82] - S.Balmez, "Generator de coduri programabile pentru testarea memoriilor MOS", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov. 1982, vol.II, pp 227-231.
5. [BALT74] - V.Baltac, Optimizarea sistemelor de operare ale calculatoarelor numerice, Facla, Timişoara, 1974.
6. [BATR81] - I.G.Bătrana, Cercetări privind structura memoriilor sistemelor de calcul interactive, Teză de doctorat, Inst.Polit.T.V.Timişoara, 1981.
7. [BELL82] - "Leak Testing in Presence of Hydrogen May Cause RAM Failures", Test and Measurement World, June 1982, pp.18.
8. [BERE83] - R.Beresford and R.J.Kozma, "Ferment and recovery stir U.S. industry to fight Japan", Electronics, Vol.56, No.13, June 3o, 1983, pp.129-142.
9. [CAPA82] - D.Căpăţînă et.al., "Translator LITEST/2 pentru echipamentul de testare automată FD-5o51", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov. 1982, vol.I, pp.285-294.
10. [CATI75] - E.Catier, "Les tests automatiques", EMI, 2o6/ 1.o6.1975.
11. [CHA377a] - J.Chase, "Computer Test Diagnostics for Memory Boards", Electronic Packing and Production, Nov. 1977.
12. [CHA377b] - J.Chase, "Computer Test Diagnostics for Memory Boards", Adar Ass. Inc., Internal Report, 1977

13. [CII72] - IRIS50, Manuel de maintenance, Tome 2, Programmes de test des organes centraux, CII, Nov. 1972.

14. [CLIF80] - R.A.Cliff, "Acceptable Testing of VLSI Components which contain Error Correctors", IEEE Trans.Comput., Vol.C-29, No.2, Feb. 1980, pp.125-134.

15. [COCK79] - D.Cocker, "An in-depth lock at MOSTEK'S high performance MK4027", in MOSTEK Data Book, 1979

16. [CRET81] - V.Creţu, Stadiul actual al dezvoltării limbajelor de programare pentru mini şi microsisteme de calcul, I.P.T.V., Timişoara, 1981, Referat de doctorat.

17. [DANC80] - M.Dancău, C.Purice, "Monitor de 1 KO pentru un sistem cu I 8085", comunicare la Primul Simpozion Naţional de Teoria Sistemelor, Craiova, 14-15 nov. 1980.

18. [DANC82a] - M.G.Dancău, R.Telescu, A.Toma, M.Cotorcă, "Sistem de operare şi editor pentru echipamente de testare automată", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov.1982, Vol.I, pp.33-38.

19. [DANC82b] - M.G.Dancău, R.Telescu, A.Boboc, A.Toma, "Limbaj de test şi interpretor pentru testoare de memorii", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov. 1982, Vol.I, pp.39-44.

20. [DANC83a] - M.G.Dancău, Probleme ale testării unităţilor de memorie RAM cu semiconductoare, I.P.T.V. Timişoara, 1983, Referat de doctorat.

21. [DANC83b] - M.G.Dancău, "Comments on Fault Location in a Semiconductor Random - Access Memory Unit", to be published in IEEE Trans.Comput.

22. [DANC83c] - M.G.Dancău, "Limbajul de test MTL - o descriere formală", comunicare la Al doilea simpozion de tehnologii şi echipamente de testare automată Cluj-Napoca, 11-12 nov. 1983.

23. [DANC83d] - M.G.Dancău, "Posibilităţi de diagnosticare "in vivo" a unităţilor de memorie RAM", comunicare

la Al doilea simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 11-12 nov.1983.

24. [DANC83e] - M.G.Dancău, R.Telescu, M.Rimbasiu, "Testarea memoriilor RAM cu semiconductoare - o privire de ansamblu", comunicare la Al doilea simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 11-12 nov. 1983.

25. [DANC83f] - M.G.Dancău, M.Ivănescu, M.Rădiţă, "O caracterizare a C.I. de memorie dinamice K565-PY3", comunicare la Al doilea simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 11-12 nov.1983.

26. [DANC83g] - M.G.Dancău, Probleme software în testarea automată a memoriilor RAM, I.P.T.V. Timişoara, 1983, Referat de doctorat.

27. [DEC77] - RSX-11M V.3.2, Utilities Procedures Manual, DEC, Maynord, MA, 1977.

28. [DUMI82] - D.Dumitraş, M.Poienar, "Limbaj specializat pentru testarea plăcilor echipate cu circuite LSI, LITEST/LSI", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj Napoca, 5-6 nov. 1982, Vol.I, pp.96-99.

29. [ELIN81] - "Orientations nouvells des circuits integrés, Electronique Industrielle, Vol.1, No.12, 15 mars. 1981, pp.21-24.

3o. [ELKI8O] - S.A.Elkind & D.P.Siewiorek, "Reliability and Performance of Error - Correcting Memory and Register Arrays", IEEE Trans.Comput., Vol.C-29, No.1o, oct.198o, pp.92o-927.

31. [FCE78] - M18 - Manual de utilizare, FCE, Bucureşti, 1978.

32. [FEE78] - W.G.Fee, "Memory Testing", in Tutorial LSI Testing, 2-nd.ed., IEEE Comput.Soc., 1978, pp.51-56.

33. [FERR79] - A.V.Ferris - Prabhu, "Improving Memory Reliability Throngh Error Correction", Computer Design, Vol.18, No.7, July 1979, pp.137-144.

34. [GHER82] - E.Gherman, "Bază de timp programabilă", comunicare la Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov. 1982.

35. [GLAN82a] - D.R.Glancy, "Computers Meet Test Equipment - and ATE is Born", Test & Measurement World, May 1982.

36. [GLAN82b] - D.R.Glancy, "Sistems ATE, part.II. Exercising the Buy option", Test and Measurement World, June 1982, pp.28-32.

37. [GLAN82c] - D.R.Glancy, "The Present and Future of VLSI and ATE", Test and Measurement World, oct.1982, pp.17-24.

38. [GLAN82d] - D.R.Glancy, "ATE Software improves Man/Machine Interface", Test and Measurement World, Nov. 1982, pp.19-3o.

39. [GREE82] - A.M.Greenspan, "Testing as a System", Test and Measurement World, June 1982, pp.8-9.

4o. [GROS81] - F.Grosvalet, "Les fabricants de mémoires MOS tournent vers la redondance", Electronique industrielle, Vol.1, No.13, Avril. 1981, pp.45-47.

41. [HART73] - R.W.Hartenstein, "Elemente der Prüftechnik für digitale Schaltungen", Elektronik, No.4, 1973, pp.117-121.

42. [HAYE75] J.P.Hayes, "Detection of Pattern Sensitive Faults in Random - Access Memories", IEEE Trans. Comput., Vol.C-24, No.2, Feb. 1975, pp.15o-157.

43. [HAYE80] - J.P.Hayes , "Testing Memories for Single - Cell Pattern - Sensitive Faults", IEEE Trans.Comput., Vol.C-29, No.3, March 198o, pp.249-254.

44. [HILL80] - F.J.Hill şi G.R.Peterson, Calculatoare numerice, Hardware - structură şi proiectare, Ed.Tehnică, Bucureşti, 198o.

45. [HP82] - "Quality of Domestic RAMs Approaching Japonese Device Reliability", Test and Measurement World, Vol.2, No.2, Feb. 1982, pp.19.

46. [IBM82] - "IBM Report Recommends Best 64K DRAM Test Patterns", in Test and Measurement World, oct.1982 pp.11.

47. [IEEE78] - IEEE Standard, ATLAS Test Language, ANSI/ IEEE Std. 416 - 1978, IEEE Inc., New York,1978.

48. [INTE76] - ISIS II User's Guide,Intel Corp.,Santa Clara, CA,1976,Manual nb.9800306D.

49. [ITC **] - "Program de testare a memoriei calculatorului I-100 DZKMA, ITC Bucureşti. Livrat în lotul de teste hard pentru calculatorul I-100.

5o. [KNAI77a] - J.Knaizuk, jr. & C.R.P.Hartmann, "An Algorithm for Testing Random - Access Memories", IEEE Trans.Comput., Vol.C-26, No.4, April 1977, pp.414-416.

51. [KNAI77b] - J.Knaizuk, jr. & C.R.P.Hartmann, "An Optimal Algorithm for Testing Stuck - at Faults in Random - Access Memories", IEEE Trans.Comput., Vol.C-26, No.11, Nov.1977, pp.1141-1144.

52. [KOPP76] - R.J.Koppel & I.Maltz, "Predicting the real costs of semiconductor memory systems", Electronics, Vol.49, 25, nov. 1976, pp.117-122.

53. [KOZM83] - R.J.Kozma, et.al., "World Market Surrey & Forecast", Electronics, Vol.56, No.1, Jan. 13, 1983, pp.125-156.

54. [LINE83] - J.R.Lineback, "MOSTEK offers 32-K-by-8 RAM", Electronics, Vol.56, No.17, 25 Aug. 1983, pp. 49-5o.

55. [LUPU82] - I.Lupu, F.Covaciu, Gh.Mureşan, "Elaborarea programelor aplicative pentru echipamente de testare automată", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, Vol.I, 1982, pp.3o3-3o6.

56. [LYMA83] - J.Lyman, "Moving probes replace Bed of nails", Electronics, Vol.56, No.13, June 3o, 1983, pp. 177.

57. [MACR77] - "Selecting test patterns for 4K RAMs", A.N. 139, Macrodata Corp., Doc. No.10006539, Woodland Hills, CA, April 1977.

58. [MACR78] - M.D.-207 Test System and Application, Macrodata Corp., Woodland Hills, CA, 1978.

59. [MART75] - R.R.Martin & H.D.Frankel, "Electronic Disks in the 1980's", Computer, Vol.8, No.2, Feb. 1975, pp.24-3o.

6o. [MILN8O] - W.Milnor & B.Johnson, "RAM maker's system tests ensure quality, free users", Electronics, Vol.53 No.26, Dec. 4, 198o, pp.142-144.

61. [MIRO82a] - C.Miron, A.Grieber, R.Vaida, "Manual LITEST I îmbunătăţit", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 1982, Vol.I, pp.29-31.

62. [MIRO82b] - C.Miron, A.Grieber, R.Vaida, "Programe de testare automată pentru plăci echipate cu circuite difitale", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 1982, Vol.I, pp.31-32.

63. [MONT75] - J.Montois, "Séquences de test pour mémories à semiconducteurs", EMI, 266/1-6-1975.

64. [NAGY82] - S.Nagy, F.Negru, L.Patakfalvi, "Un sistem software pentru ETA numerice", in Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov.1982, vol.I, pp. 45-53.

65. [NAIR78] - R.Nair, S.M.Thathe & J.A.Abraham, "Efficient Algorithms for Testing Semiconductor Random - Access Memories", IEEE Trans.Comput., Vol.C-27, No.6, June 1978, pp.572-576.

66. [NAIR79] - R.Nair, "Comments on An Optimal Algorithm for Testing Stuck-at Faults in Random - Access Memories", IEEE Trans.Comput., Vol.C-28, No.3, March 1979, pp.258-261.

67. [PATA81] - L.Patakfalvi, Manual de programare în limbajul LITEST 2, IPA, Cluj-Napoca, 1981.

68. [PATA82] - L.Patakfalvi, "LITEST- 2 - limbaj de programare evaluat pentru testoarele din familia THETA", comunicare la Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov. 1982.

69. [PENE72] - W.M.Peney & L.Lau, eds., MOS Integrated Circuits, Van Nostrand-Reinhold, New York, 1972.

7o. [PHIL8O] - C.M.Philpott, "Testing Logic on Memory Boards" Micro Control Comp., Publication 00048, Minneapolis, N.M., 198o.

71. [POSA79] - J.G.Posa, "Programming microcomputer systems with highlevel languages", Electronics, Jan. 18, 1979, pp.1o5-112.

72. [POSA81a] - J.G.Posa, "Soviet chips feature refined fabrication but mimic U.S.ICs", Electronics, Vol.54, No.8, Jan.81, pp.39-4o.

73. [POSA81b] - J.G.Posa, "What to do when the bits go out", Electronics, Vol.54, No.15, 28 July 1981, pp. 117-12o.

74. [PRED82] - A.Predoi, "Ingineria asistată de calculator", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov. 1982, vol.I, pp.11-17.

75. [PROE79] - R.Proebsting, "Dynamic MOS RAMs", in MOSTEK Data Book, 1979.

76. [OWEN79a] - R.W.Owen, "Optimized testing of 16K RAMs", in MOSTEK Data Book, 1979.

77. [OWEN79b] - R.W.Owen, "A Testing philosophy for 16K dynamic memories", in MOSTEK Data Book, 1979.

78. [OWEN79c] - R.W.Owen, "Addressing consideration when testing the MK4027", in MOSTEK Data Book, 1979.

79. [RIMB82] - M.Rimbaşiu, S.Balmez, "Evoluţia aparaturii de testare a memoriilor MOS realizate la ITC Timişoara", comunicare la Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov.1982.

8o. [SCHE81] - F.A.Scherpenberg, "Static n-MOS RAM idles on trickle current", Electronics, Vol.54, No.2, 27 Jan. 1981, pp.129-132.

81. [SCHL78a] - Le Logiciel BASCH, Enertec-Schlumberger, St. Etienne, France, 1978.

82. [SCHL78b] - Le Logiciel Basch Instructions programmant les appareils du sistème, Enertec-Schlumberger, St.Etienne, France, 1978.

83. [SCHW81] - Ph.Schwartz, "Le defi des années 8o : maintenir la productivité des testeurs LSI et VLSI", Elec-

tronique industrielle, Vol.1, No.13, Avril.1981, pp.XXXVII-XLVII.

84. [SETH81] - S.C.Seth & K.Narayanaswamy, "A Graph Model for Pattern Sensitive Faults in Random-Access Memories", IEEE Trans.Comput., Vol.C-3o, No.12, Dec. 1981, pp.973-977.

85. [SIRB82] - M.Sîrbu, "Testarea asistată de calculator, un procedeu tehnologic industrial, actual şi de perspectivă", în Primul simpozion de tehnologii şi echipamente de testare automată, Cluj-Napoca, 5-6 nov.1982, vol.I, pp.18-25.

86. [SMAY83] - M.C.Smayling & M.Maekwa, "256-K dynamic RAM is more than just an upgrade", Electronics, Vol.56, No.17, 25 aug. 1983, pp.135-137.

87. [SMIT81] - R.T.Smith, "Using a laser beam to substitute good cells for bad", Electronics, Vol.54, No.15, 28 July 1981, pp.131-134.

88. [SRIN78] V.D.Srini, "Fault location in a Semiconductor Random-Access Memory Unit", IEEE Trans.Comput., Vol.C-27, No.4, April 1978, pp.349-358.

89. [STON79] - P.Stone & J.F.McDermid, "Circuit Board Testing: Cost Effective Production Test and Troubleshooting, H.P.Journal, Vol.3o, No.3, March 1979,pp.2-8.

9o. [SUK8O] - D.S.Suk & S.M.Reddy, "Test Procedures for a Class of Pattern Sensitive Faults in Semiconductor Random-Access Memories", IEEE Trans.Comput., Vol.C-29, No.6, June 198o, pp.419-429.

91. [SUK81] - D.S.Suk & S.M.Reddy, "A March Test for Functional Faults in Semiconductor RAMs", IEEE Trans. Comput., Vol.C-3o, No.12, Dec.1981, pp.982-986.

92. [SUL81] - R.Sul and K.C.Hardee, "Designing static RAMs for yield as well as speed", Electronics, Vol.54, No.15, 28 July 1981, pp.212-126.

93. [TEKT74] - "Efficient testing of MOS 4K RAMs", Tektronix System Application Note 70K4.0, 1974.

94. [TELE82a] - R.Telescu, M.G.Dancău, MOSTEST-03D - Manual de utilizare, ITC, Timişoara, 1982.

95. [TELE82b] - R.Telescu, D.Vlăduţiu, M.Cotarcă, M.G.Dancău, "Microcalculator pe două plachete orientat pe sistemul de test", în Primul simpozion de tehnologii şi echipamente de testare automată", Cluj-Napoca, 5-6 nov.1982, vol.I, pp.12o-125.

96. [VLAD82] - M.Vlăduţiu, Creşterea coeficientului de disponibilitate al echipamentelor automate de prelucrare a informaţiei prin metode de testare neconvenţionale, Teză de doctorat, I.P. Bucureşti, 1982.

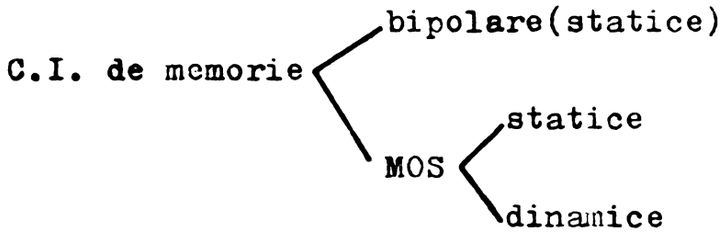
97. [WHIT83] - A.Whiteside, "VLSI test system grows in pin count and functionality", Electronics, Vol.56, No.11, May 31, 1983, pp.155-16o.

98. [ZIL077] - MCS 1. 0, Software User's Manual/preliminary/ Zilog, Cuppertino, CA, March 1977.

A N E X A A.1

CIRCUITE INTEGRATE DE MEMORIE RAM
STRUCTURA SI ORGANIZARE

După tehnologia în care sînt realizate celulele de memorare, CI de memorie RAM se pot clasifica în bipolare şi MOS. La rîndul lor, C.I. MOS se împart în statice şi dinamice după structura celulei de memorare.



A.1.1. Circuite de memorie statice

A.1.1.1. Celula de memorare în tehnologie bipolară

Schema tipică a unei celule bipolare de memorare este prezentată în fig.A1.1.

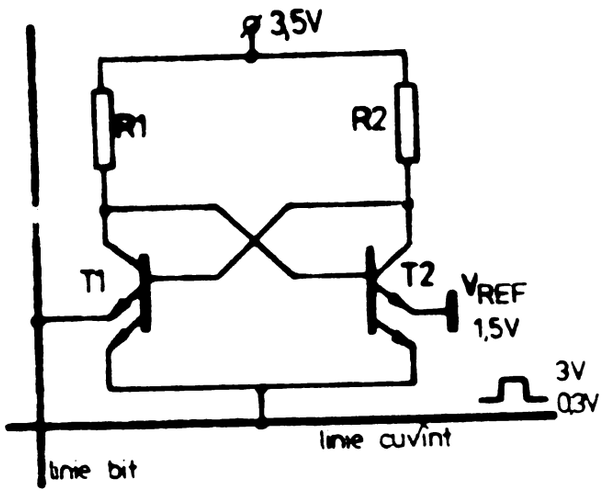


Fig.A1.1. Element de memorie bipolară

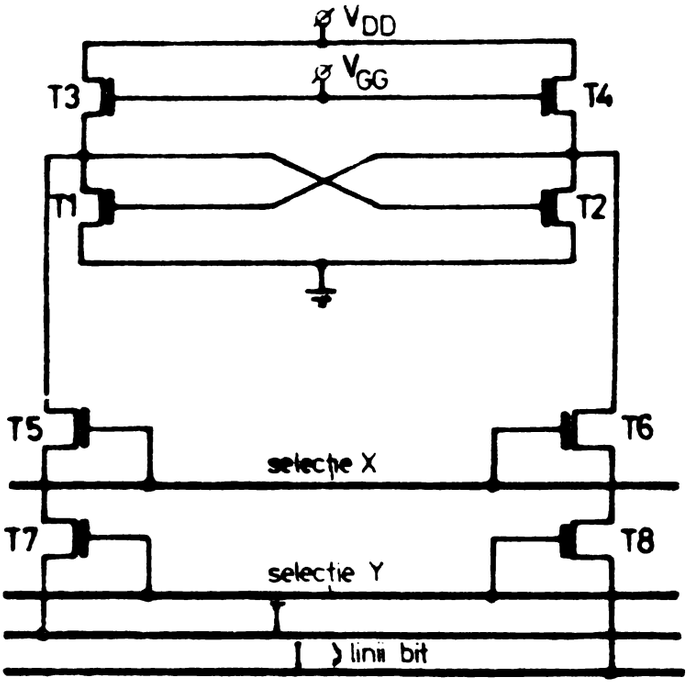
Ea este formată în esenţă dintr-un bistabil cu două tranzistoare. Descrierea funcţionării ei poate fi găsită în [HILL80, pp.64-68]

A.1.1.2 Celula de memorare statică MOS

Pe măsura evoluţiei tehnologiilor MOS au fost utilizate diverse scheme pentru celula de memorare statică.

In fig.A1.2 este prezentată schema unei celule cu 6

tranzistoare. Tranzistoarele T_3 şi T_4 au rolul de rezistenţe de drenă. Bistabilul de memorare este format din tranzistoarele T_1 şi T_2.



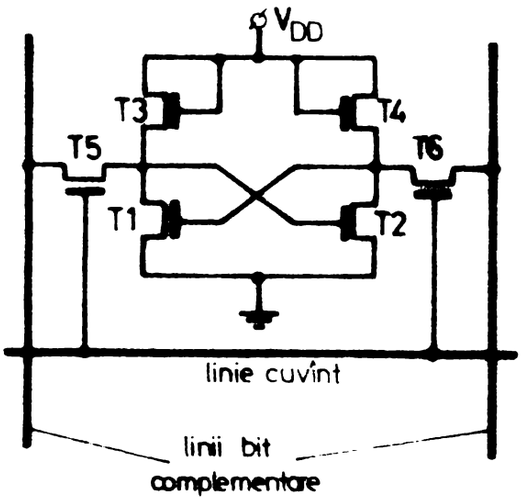
A1.2. Celula de memorie statică cu 8 tranzistoare MOS

Caracteristic celulelor statice este prezenţa a două linii de bit complementare.

Selecţia celulelor se face prin aplicarea semnalelor logice 1 (logică negativă) pe cele 2 linii de selecţie. Aceasta are ca efect conectarea drenelor tranzistoarelor T_1 şi T_2 ale bistabilului la liniile de bit.

Inscrierea se face prin forţarea liniilor de bit în starea dorită.

Figura A1.3 prezintă o celulă cu 6 tranzistoare, a cărei selecţie se face printr-o singură linie de cuvînt, şi care ocupă o arie mai mică.

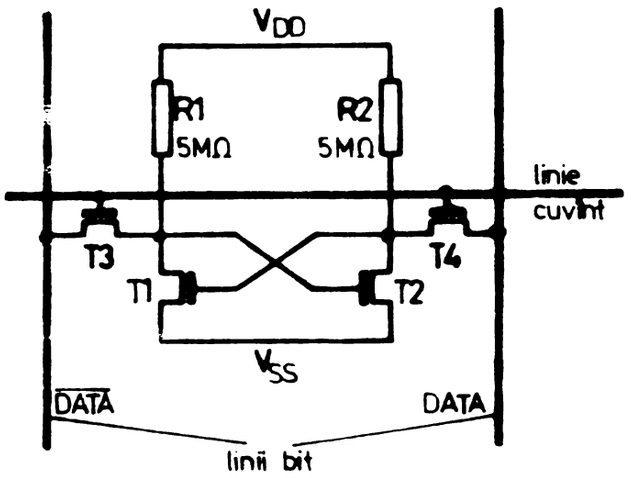


A.1.3. Celula de memorie statică cu 6 tranzistoare MOS

Prin activarea liniei de cuvînt, tranzistoarele T5 şi T6 conectează celula la liniile de bit [PEN72].

La celulele de mai sus rezistenţele de drenă erau realizate cu tranzistoare MOS deoarece acestea ocupau o arie mai mică.

Pe măsură ce evoluţia tehnologiei a permis realizarea de rezistenţe de valori mari şi de arie mică, acestea au înlocuit tranzistoarele MOS ca rezistenţe de drenă.



A.1.4. Celulă statică cu 4 tranzistoare MOS.

Firma MOSTEK prezintă o celulă (fig.A1.4) vînd rezistenţe din polisiliciu, valorile rezistenţelor fiind de 5M Ω.

Recent a fost menţionat în literatură [SCHE81] un C.I. de memorie MK48D02, 2 k x 8 bit avînd rezistenţele de drenă de valori de $10^{10}\Omega$, ceea ce permite reducerea semnificativă a curentului consumat.

A.1.1.3. Organizarea memoriilor RAM statice

Celulele de memorare statice sînt organizate sub formă de matrice. Se va prezenta organizarea memoriei 2147 de 4 k x 1 bit (fig.A1.5).

După cum se poate observa, adresa reală în matrice a unei celule nu coincide cu adresa externă, fiind necesară o renumerotare a liniilor de adresă.

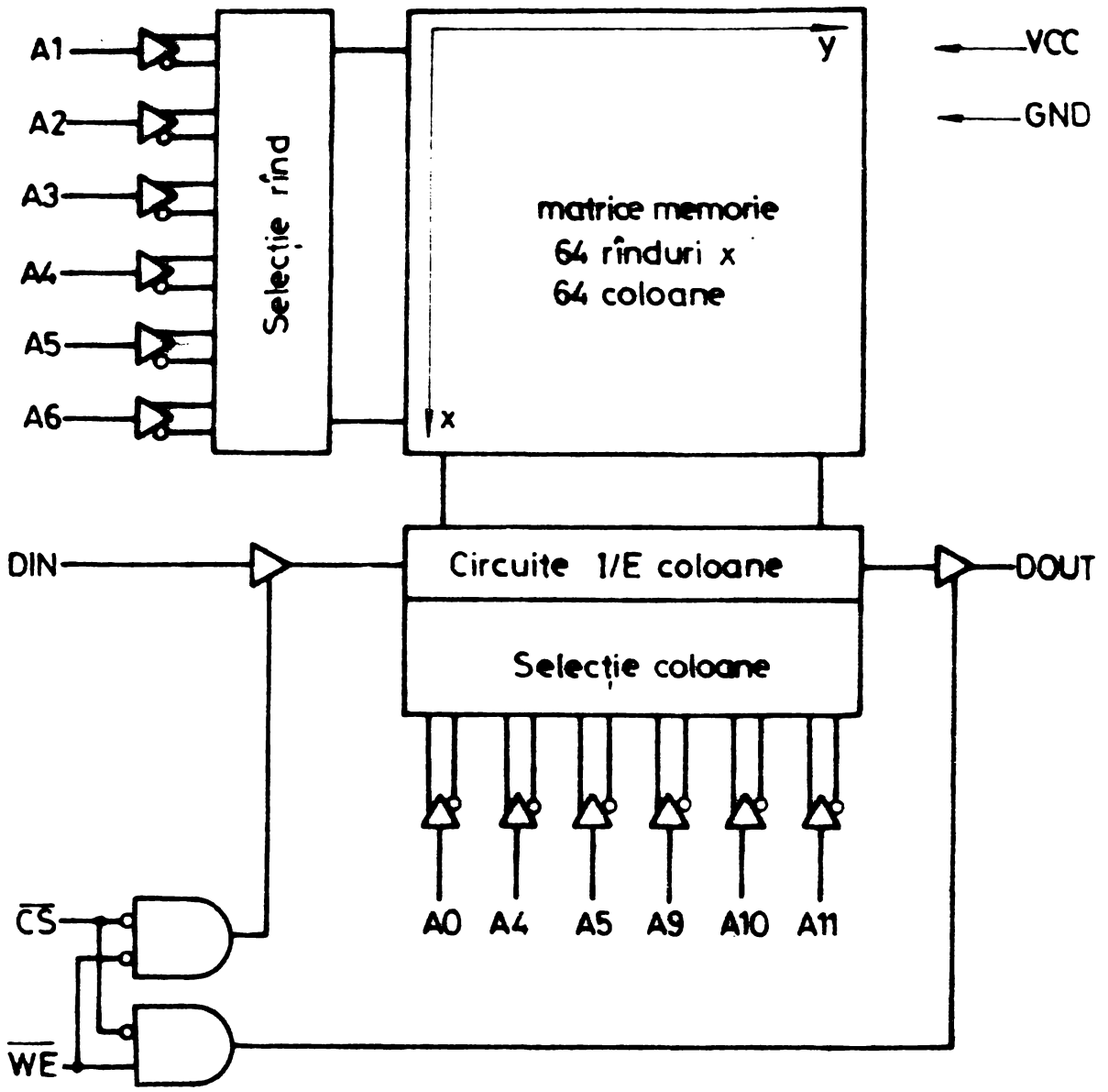
Acesta este un caz simplu, în care fiecare bit al adresei internă este funcţie de un singur bit al adresei externe.

A.1.2. C.I. de memorie RAM dinamice

Circuitele integrate de memorie dinamice au apărut din necesitatea de a realiza capacităţi mai mari pe aceeaşi pastilă de siliciu, aria ocupată de o celulă dinamică fiind mai mică decît cea ocupată de o celulă statică realizată în aceeaşi tehnologie.

A.1.2.1. Celula de memorare dinamică

Spre deosebire de celula statică, elementul esenţial al celulei dinamice îl constituie o capacitate, memorarea făcîndu-se prin încărcarea acestei capacităţi fie la tensiunea



A.1.5. Schema bloc a memoriei 2147

de alimentare, fie la o tensiune nulă. Datorită curenţilor de scurgere ai capacităţii şi a rezistenţei finite în stare blocată a tranzistoarelor de acces, memorarea este limitată în timp. Este necesară reîncărcarea periodică a capacităţii fiecărei celule de memorare.

Primele celule dinamice erau formate din trei tranzistoare (fig.A1.6).

Pentru înscriere, prin activarea liniei de scriere cuvînt T1 trece în conducţie iar tensiunea pe capacitate este adusă la valoarea tensiunii de pe linia de (scriere) bit.

Tensiunea pe capacitate constituie tensiunea grilă-dren a tranzistorului T2. La citire, prin activarea liniei de citire cuvînt T3 trece în conducţie. Funcţie de valoarea

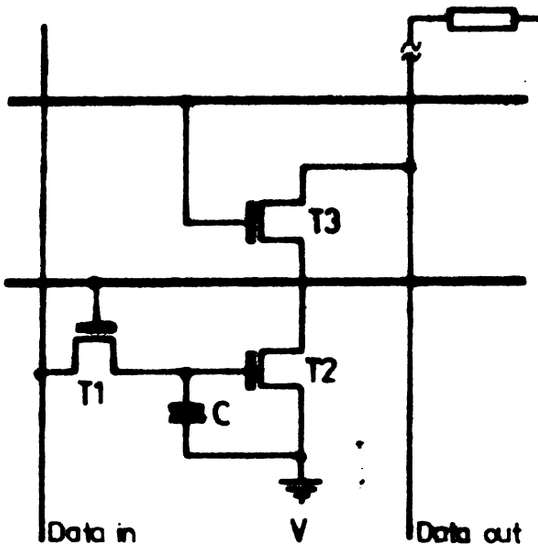


Fig.Al.6a. Celula dinamică a C.I. INTEL 1103

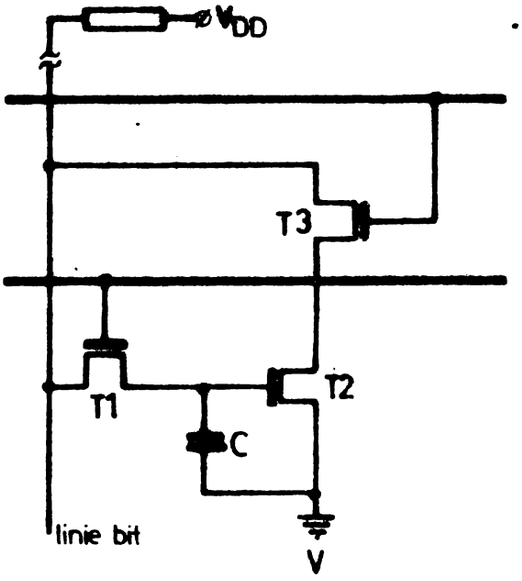


Fig.Al.6b. Celula dinamică a C.I. MOSTEK 4006

tensiunii pe capacitatea de memorare T2 este în conducţie sau blocat şi tensiunea de pe linia de (citire) bit este fie 0 (T2 în conducţie) fie V_{DD} (T2 blocat). Diferenţa dintre tensiunile de pe linia de bit în cazul citirii de "0" sau "1" este de ordinul tensiunii de alimentare V_{DD} ca şi în cazul memoriilor statice. Citirea este nedistructivă, în sensul că sarcina acumulată pe capacitate nu este alterată prin citire.

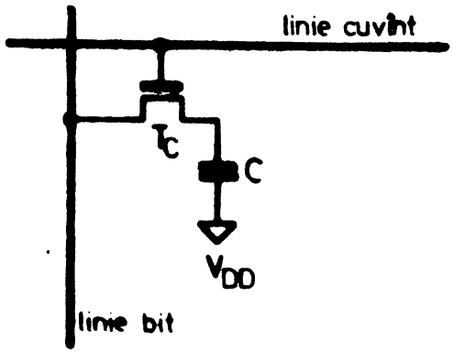


Fig.Al.7. Celula dinamică cu un tranzistor

Evoluţia tehnologiei NMOS a permis trecerea la celule de memorare cu un singur tranzistor, (fig. Al.7) aria acestei celule fiind mai puţin de jumătate din aria celulei cu trei tranzistoare. Astfel aria celulei din C.I. MK4027 (4 k x 1) este de 1.oo8 mil^2 ($6.5xlo^{-4}mm^2$) iar aria celulei din circuitul MK4116 (16 k x 1) este de o,55 mil^2 ($3.55xlo^{-4}mm^2$). Circuitele actuale au aria celulei şi mai redusă.

Funcţionarea acestei celule este descrisă în [PROE79] . Citirea celulei se face prin trecerea în stare de conducţie a

tranzistorului T_c. Are loc o redistribuire a sarcinii între capacitatea de memorare şi capacitatea liniei de bit. Aceasta din urmă este mult mai mare (cca 1 pF) faţă de cea a celulei (cca o,o4 pF) şi semnalele culese pe linia de bit sînt de ordinul zecilor sau sutelor de mV. Citirea este distructivă.

Pentru a micşora capacitatea liniei de bit a fost necesară reducerea lungimii acestei linii şi introducerea de amplificatoare de citire.

Se va prezenta în continuare organizarea şi funcţionarea C.I. de memorie la care matricea de celule de memorare este divizată în două submatrici complementare, deoarece aceasta este organizarea cea mai frecvent întîlnită astăzi.

A.1.2.2. Amplificatoare de citire statice

Funcţionarea acestor amplificatoare este descrisă în [PROE79]. Schema unei coloane de celule este prezentată în fig.A1.8.

Reducerea capacităţii liniei de bit s-a făcut prin împărţirea ei în două şi plasarea amplificatorului de citire la mijloc. Amplificatorul este diferenţial cu reacţie pozitivă, fiind activat de un tact intern. In continuare cele două jumătăţi ale liniei de bit vor fi denumite ca linia de bit superioară şi inferioară.

In stare de repaus, cele două linii de bit sînt echilibrate la aceeaşi tensiune. La declanşarea unui ciclu de memorie ele sînt lăsate pentru scurt timp să flo eze, apoi, în urma activării unei linii de selecţie rînd (linie cuvînt), toate celulele din rîndul selectat sînt conectate la liniile de bit superioare sau inferioare ale coloanelor respective. In fiecare coloană este selectată o singură celulă, fie din jumătatea su erioară fie din cea inferioară. In urma trecerii în stare de conducţie a tranzistorului celulei selectate are loc o redistribuire a sarcinii între capacitatea celulei şi linia de bit (superioară sau inferioară). In cele ce urmează, prin tensiunea pe celulă se va înţelege diferenţa de potenţial dintre armătura dinspre tranzistor a capacităţii şi masă.

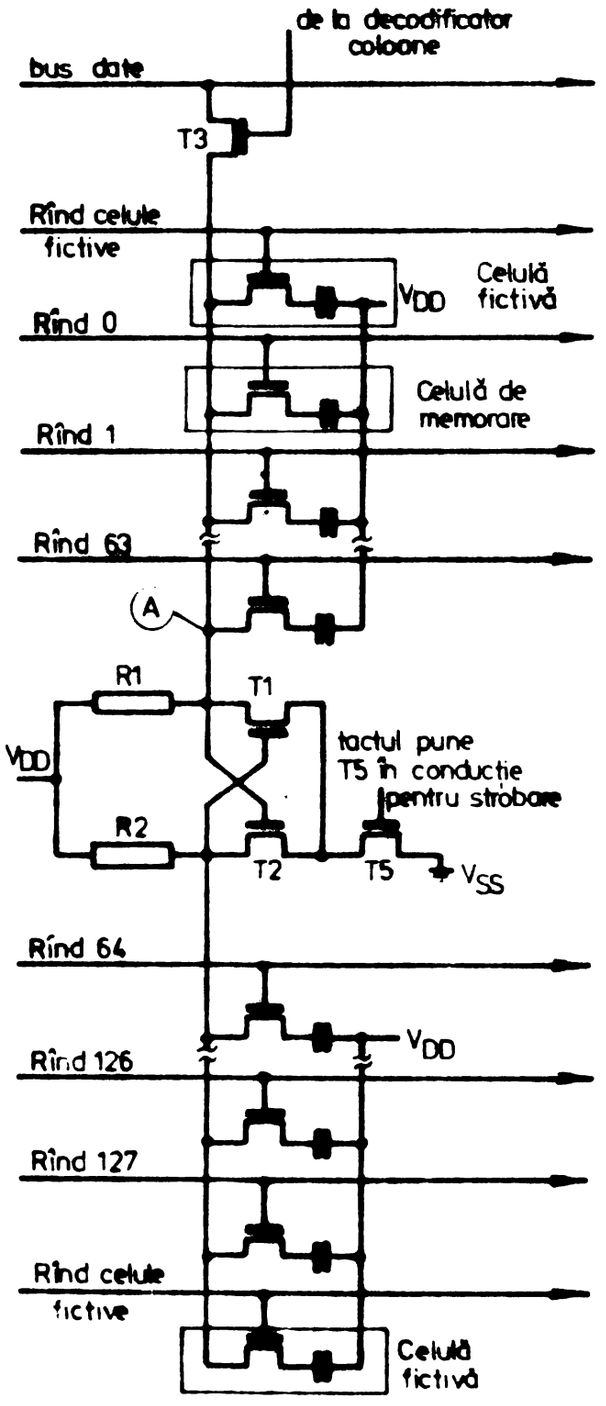


Fig.A1.8. Coloană de celule dinamice cu amplificator de citire static

Dacă tensiunea înmagazinată pe celulă a fost apropiată de 0 ("Low") tensiunea rezultată pe linia de bit va fi mai coborîtă decît în cazul în care tensiunea pe celulă a fost apropiată de V_{DD} ("high"). Diferenţa dintre cele două valori ale tensiunii rezultate pe linia de bit este mai mică de o,5 V datorită atenuării liniei de bit.

Fragmentul de linie de bit care nu are acces la celula selectată este conectat la o celulă fictivă (dummy cell) Această celulă este încărcată la o tensiune nulă în stare de repaus şi are capacitatea egală cu jumătate din capacitatea celulelor de memorare. Ca urmare, tensiunea rezultată pe acest fragment de linie de bit va fi la jumătatea intervalului dintre tensiunile "high" şi "low". Diferenţa de potenţial dintre cele două fragmente de linie de bit va fi deci mai mică decît o,25 V.

După redistribuirea sarcinii pe cele două linii de bit, tranzistorul T5 este trecut în stare de conducţie. Datorită reacţiei pozitive a bistabilului format de T1 şi T2 diferenţa

de potenţial dintre nodurile A şi B ajunge la o valoare a tensiunii de alimentare V_{DD}, nodul la care este conectată celula citită ajungînd fie la potenţialul V_{DD} fie la 0, după care tensiunea iniţială pe celulă a fost mai mare sau mai mică decît 1/2 V_{DD}. Are loc astfel reîncărcarea celulei citite la tensiunea anterioară.

Dacă celula citită face parte din coloana selectată atunci prin trecerea în conducţie a tranzistorului T3, nodul A este conectat la linia de date.

Scrierea: Se presupune că se doreşte înscrierea unei tensiuni V_{DD} în celula 64 şi că această celulă a fost anterior în starea opusă.

După echilibrarea sarcinii pe liniile de bit şi reîncărcarea celulelor în starea iniţială, nodul A se află la potenţial V_{DD}, iar nodul B la potenţial 0.

Aducerea nodului B la potenţial V_{DD} se face prin forţarea nodului A la potenţial 0, procedînd astfel basculareа bistabilului T1, T2. Ridicarea potenţialului nodului B se face numai prin rezistenţa R2, cu T2 blocat. Valori mari ale acestor rezistenţe duc la încărcarea lentă a capacităţii liniei de bit (ciclu de scriere lung) iar valori mici duc la creşterea excesivă a puterii disipate pe pastilă. Puterea disipată pe rezistenţele R1 şi R2 din toate amplificatoarele de citire este de obicei 50% din puterea totală disipată [PROE79].

Pentru a elimina acest inconvenient şi amplificatorul de citire a fost realizat dinamic.

A.1.2.3. Amplificatoarele de citire dinamice

Schema unui astfel de amplificator este dată în fig.9, iar funcţionarea sa este descrisă în [OWEN79a]. In stare de repaus cele două linii de bit sînt preîncărcate la V_{DD} iar celulele fictive la 0.

La iniţierea unui ciclu de memorie, celula selectată este conectată la fragmentul său de linie de bit iar linia de bit complementară este conectată la celula fictivă. In urma redistribuirii de sarcină tensiunea pe linia de bit conecta-

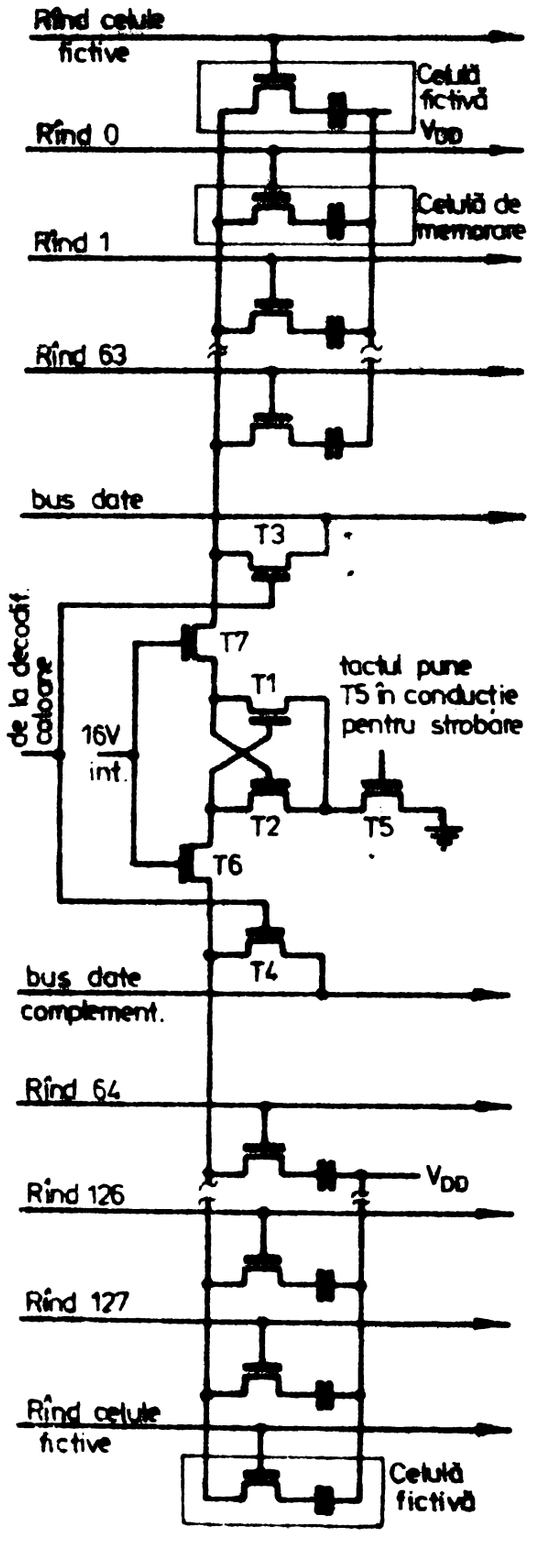


Fig.A1.9. Coloană de celule dinamice cu amplificator de citire dinamic

tă la celula fictivă va fi V_{DD}--o,4V. Tensiunea pe linia de bit conectată la celula selectată va fi cea V_{DD}-o,3V dacă tensiunea pe celulă a fost "high" şi de cca V_{DD}-o,5V dacă tensiunea pe celulă a fost "low"

După redistribuirea sarcinii, T5 este trecut în stare de conducţie, activînd reacţia pozitivă a bistabilului din T1 şi T2. Datorită reacţiei pozitive, fragmentul de linie de bit care a avut o tensiune mai mică în urma citirii va fi forţat la o tensiune 0. Astfel dacă tensiunea pe celula citită a fost mai mică decît 1/2× V_{DD}, în urma activării bistabilului T1, T2 celula va fi încărcată la o tensiune nulă. Dacă tensiunea pe celula citită a fost mai mare decît 1/2 • V_{DD}, tensiunea rezultată pe ea va fi mai mare decît V_{DD}-o,4V. Se observă că prin citirea rîndului respectiv de celule, toate celulele rîndului sînt regenerate.

Scrierea: Pentru înscriere a fost necesară introducerea liniei de date complementare. La înscriere, se forţează în nodul A informaţia dorită iar în nodul B informaţia complementară. Linia de date complementară îndeplineşte în acest caz rolul rezistenţei R2. Nu apare un consum suplimentar de curent pentru for-

ţarea bistabilului T1, T2 deoarece acesta nu are rezistenţe de drenă.

Două defecte sînt specifice acestor tipuri de amplificatoare de citire. Primul apare în cazul echilibrării imperfecte ale celor două fragmente de linii de bit, potenţialul lor iniţial depinzînd de potenţialul din ciclul precedent. Acesta face ca rezultatul unei citiri să depindă de rezultatul citirii din ciclul anterior. In cazul unei serii de citiri succesive care duc amplificatorul în aceeaşi stare (de pildă nodul A - "low" şi nodul B - "high", dezechilibrul se cumulează şi există posibilitatea ca amplificatorul să interpreteze greşit o citire care ar trebui să-l ducă în starea opusă.

Al doilea tip de defect, constînd în asimetria ansamblului format de amplificator, linii de bit şi celule fictive, se manifestă cu precădere atunci cînd între încărcarea şi citirea unei celule este modificată tensiunea de alimentare V_{DD}. De exemplu, o celulă este încărcată la tensiunea V_{DDmin}. După expirarea timpului de regenerare, tensiunea pe celulă va fi $1/2\ V_{DDmin} + U_c$. Citirea se face practic prin compararea cu o tensiune de referinţă $U_{ref} = \frac{1}{2} V_{DD}$. Dacă citirea se face avînd circuitul alimentat la V_{DDmax} atunci există posibilitatea ca

$$U_c = \frac{1}{2} V_{DDmin} + \quad U_c < \frac{1}{2} V_{DDmax}$$

şi conţinutul celulei este interpretat ca fiind starea complementară.

Este necesară precizarea că este vorba de citirea internă a celului respective. In orice ciclu de memorie sînt citite intern toate celulele dintr-un rînd.

A.1.2.4. Organizarea C.I. de memorie dinamice

După cum s-a arătat mai sus, este avantajoasă utilizarea celulelor de memorare cu un singur tranzistor şi amplificatoare diferenţiale de citire. Aceasta duce la împărţirea matricii de celule în două jumătăţi una dintre ele memorînd informaţia în logică negativă.

Firme care nu utilizează amplificatoare dinamice, au împărţit chiar în 4 matricea pentru scurtarea liniilor de bit In fig.A.1.lo, este prezentată structura memoriei INTEL 2116 (16Kx1). [OWEN79b].

O structură cu două matrici de memorare şi amplificatoare de citire dinamice este cea a memoriei MK 4116 (MOSTEK) [OWEN79b] prezentată în fig.A1.11. De menţionat că şi C.I. de memorie fabricate în URSS au aceeaşi structură [POSA81a].

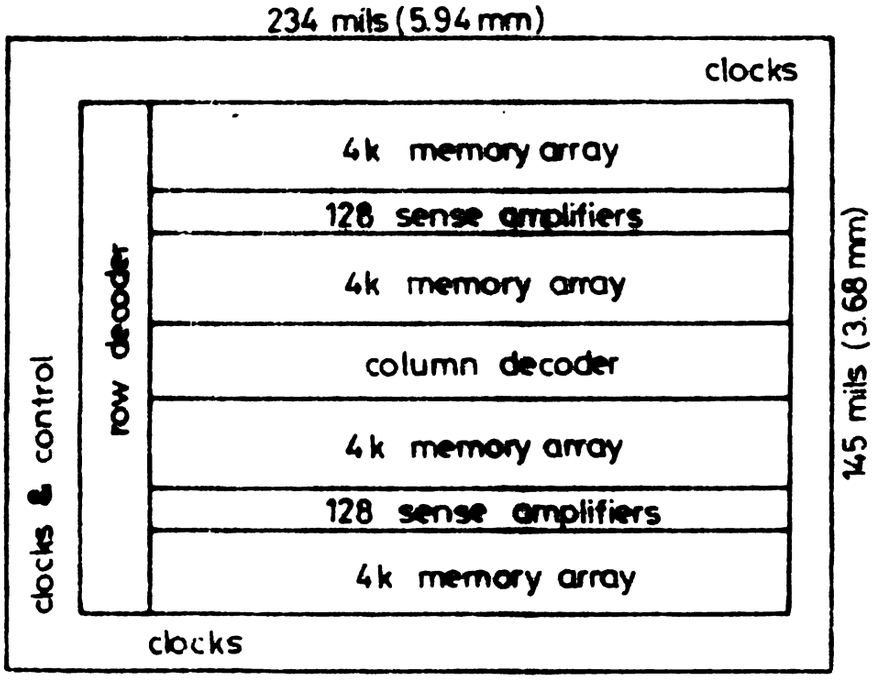


Fig.A1.lo. Structura memoriei INTEL 2116

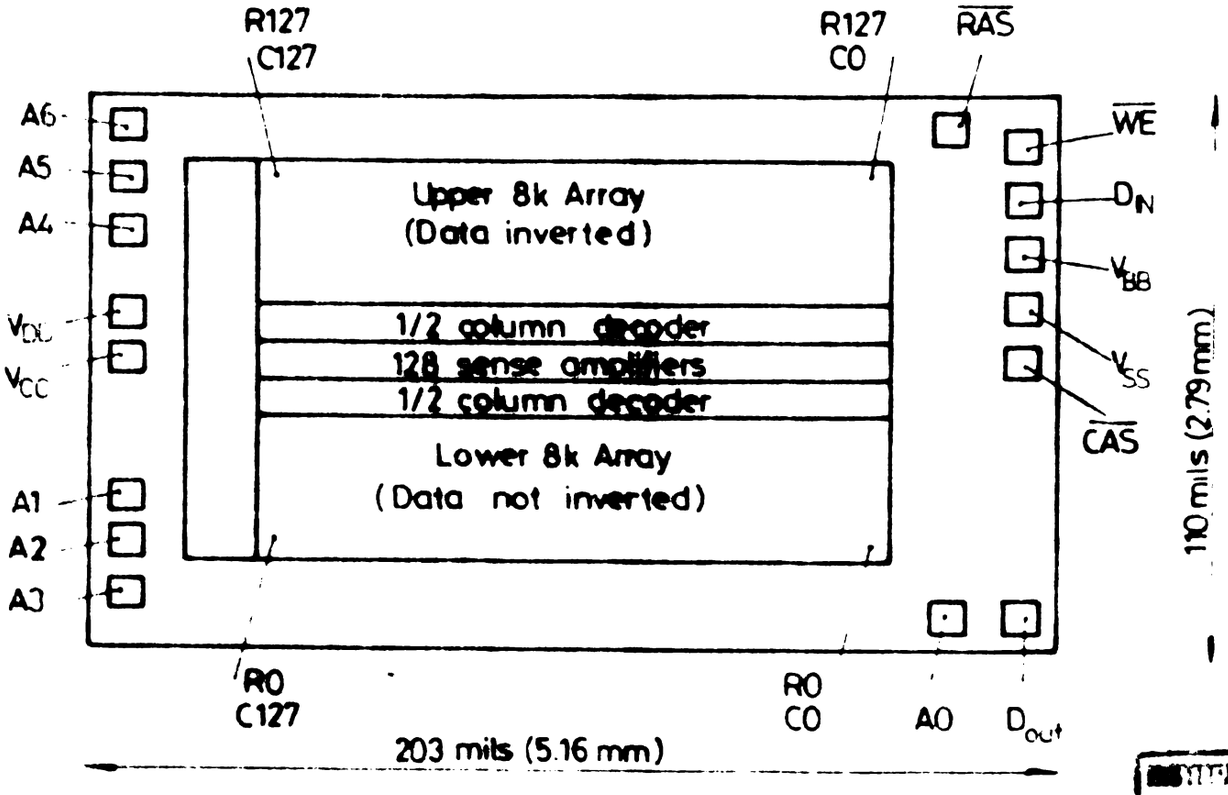


Fig.A1.11 Structura memoriei MOSTEK MK 4116

Si la C.I. de memorie dinamice adresa internă a unei celule este diferită de adresa externă. Mai mult, fiecare bit al adresei interne poate fi funcţie de mai mulţi biţi ai adresei externe. Deoarece mulţi algoritmi de test verifică sensibilităţi între celule fizic adiacente, este necesară adresarea topologică. Înţelegîndu-se prin aceasta că circuitului testat nu i se aplică cuvîntul de adresă Y,X generat de dispozitivul de test ci o transformată a acestuia, în aşa fel încît să fie adresată celula cu adresa internă Y,X.

[COCK79] indică transformarea necesară pentru C.I. de memorie MK4o27 (fig.A1.12).

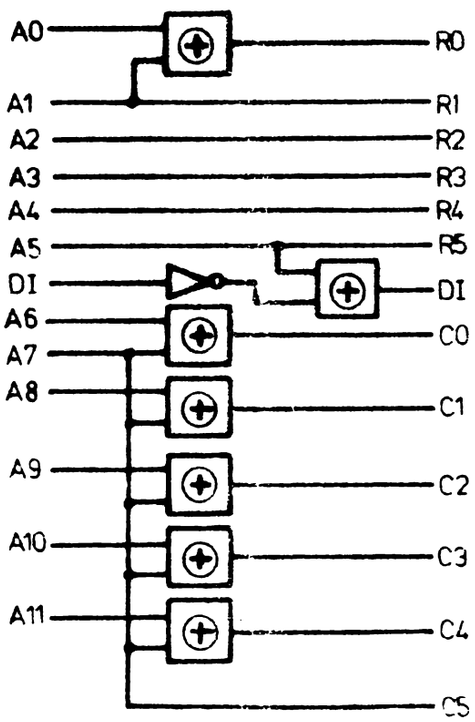


Fig.A1.12. Transformările necesare pentru adresarea topologică în C.I. MOSTEK MK 4o27

Pentru memoria MK 4116 [OWEN79c] indică următoarea metodologie. In primul rînd este necesară o rememorare a pinilor de adresă, conform tabelului de mai jos.

Pentru o adresare topologică este necesară apoi transformarea adreselor conform schemei din fig.A1.13.

Problema adresării topologice este mai complexă la actualele C.I. de foarte mare capacitate (64K, 256K). La aceste C.I. există rînduri şi coloane de celule suplimentare cu care

Nr.pin	Denumire în foaia de catalog	Funcţie reală
13	A6	A0
1o	A5	A1
11	A4	A2
12	A3	A3
7	A1	A4
6	A2	A5
5	A0	A6

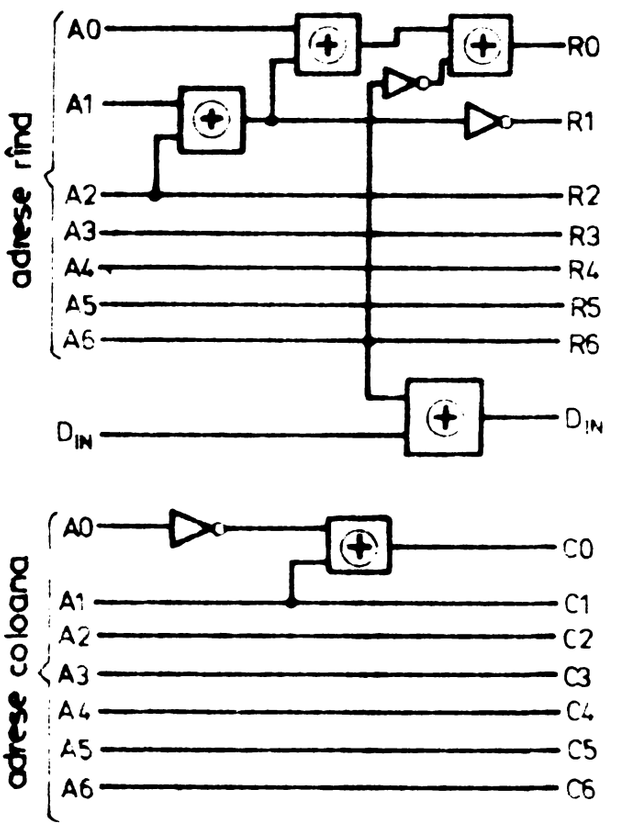


Fig.A1.13. Transformările necesare pentru adresarea topologică în C.I. MOSTEK MK 4116

se înlocuiesc rîndurile (coloanele în care s-au descoperit defecte în cursul testării pastilei la producător). Se obţine astfel o scădere spectaculoasă a numărului de pastile rebut însă topologia internă diferă de la o pastilă la alta iar timpul de acces creşte cu cca. 5 ns în cazul utilizării coloanelor (rîndurilor suplimentare) [POSA01b], [SUL01], [ABSO01]. Conectarea rîndurilor (coloanelor suplimentare), în locul celor defecte (reconfigurarea decodificatoarelor) se face prin

fuzibile care sînt programate în cursul testării pastilei sau cu ajutorul laserului. O comparaţie între cele două metode este făcută de [SMITH81]. Firma MOSTEK a prevăzut la un C.I. static 16K x 1 o funcţie "roll-call" [POSA81b]. Pentru a activa această funcţie se activează într-un nod nestandard semnalele de control. Ieşirea C.I. va fi "1" numai în cazul în care coloana adresată a fost înlocuită cu una din coloanele de rezervă.

La ora actuală (25 aug.1983) C.I. de 64K de memorie RAM sînt produse în mod curent, atît în varianta 64K x 1 cît şi 16K x 4. Mai multe firme au prezentat deja prototipuri sau eşantioane de C.I. de 256K dar numai T.I. şi MOSTEK le oferă în producţie de serie, deocamdată în configuraţia 64K x 4 (T.I.) şi 32K x 8 (MOSTEK) [LINE83]. Aceste configuraţii sînt mai uşor testabile decît cea de 256K x 1. Modificările tehnologice introduse în fabricaţia memoriilor odată cu C.I. de 256K RAM sînt prezentate în [SMAY83].

A.1.3. Concluzii

Circuitele integrate de memorie RAM statice sînt simplu de utilizat, schemele pentru comanda lor fiind compuse doar din circuite combinaţionale. Densitatea acestor circuite este mai mică decît a celor dinamice. Sub aspectul preţului, un C.I. de memorie static este echivalent cu un C.I. de memorie dinamic cu o capacitate de 4 ori mai mare.

Circuitele dinamice permit o densitate foarte mare, existînd astăzi circuite de 64Kilobiţi. Schemele de comandă sînt complexe, în special datorită faptului că fiecare rînd de celule trebuie regenerat la un interval de 2 ms. Se reduce astfel coeficientul de disponibilitate cu 2 - 5%. Ele sînt însă avantajoase sub aspectul preţului şi al puterii consumate. Sînt recomandate de aceea pentru unităţi de memorie de capacitate mare.

A N E X A A.2

TESTE STANDARD PENTRU MEMORII RAM SEMICONDUCTOARE

Algoritmii de test s-au dezvoltat odată cu evoluţia memoriilor semiconductoare. In cele ce urmează se va face o prezentare a lor şi a defectelor pe care le pot detecta.

Memoria poate fi privită ca o maşină secvenţială avînd 2^N stări [HAYE75]. Procedurile de test bazate pe metodele de identificare a maşinii secvenţiale, de tipul [HAYE75] pot pune în evidenţă toate defectele teoretic detectabile dar aplicarea lor practică este exclusă datorită numărului mare de stări. In testarea memoriilor se utilizează proceduri algoritmice, care pot fi clasificate în proceduri "ad hoc" [VLAD82] pentru detectarea anumitor clase de defecte şi proceduri elaborate pe baza unor modele de defecte mai restrictive decît modelul Hayes([KNAI77a], [KNAI77b], [NAIR78], [NAIR79], [SUK81]). Aceste proceduri algoritmice vor fi prezentate în continuare.

W.G.Fee [FEE78] consideră că pentru memoriile dinamice cu organizare 4o96 x 1 următoarele defecte posibile trebuie verificate

Unicitatea adresei
Viteza decodificatoarelor de adresă
Perturbări ale celulei
Perturbări ale celulelor adiacente
Perturbări ale coloanelor
Perturbări ale coloanelor adiacente
Perturbări ale rîndurilor
Perturbări ale rîndurilor adiacente
Sensibilitatea la informaţie
Revenirea după scriere
Regenerarea

In afară de defectele posibile de mai sus alţi autori [TEKT74], [MONT76], [MACR77] consideră că mai trebuie testate

- scrierile multiple
- revenirea amplificatoarelor de citire.

In cele ce urmează va fi explicată semnificaţia acestor defecte

Unicitatea adresei. Prin verificarea unicităţii adresei se înţelege verificarea existenţei fiecărei celule de memorare ca o entitate unică, separat adresabilă. Neîndeplinirea acestei condiţii poate fi cauzată fie de defecte ale matricii de celule de memorare, de defecte a unor amplificatoare de citire sau ale decodificatoarelor care pot avea linii blocate pe 1 sau 0 sau în scurtcircuit. Viteza de comutare a decodificatoarelor de adresă depinde atît de starea anterioară cît şi de starea următoare, în care se comută. In fig.A3.1 şi A3.2 se poate observa reducerea semnificativă a domeniului de funcţionare pentru testul GALPAT a cărui secvenţă de adresare conţine toate salturile de adresă posibile.

Prin perturbări ale celulei se înţeleg defectele de alterare a informaţiei conţinute de o celulă prin operaţii de scriere/citire efectuate asupra acelei celule.

Prin perturbări ale celulelor adiacente , ale coloanelor ale coloanelor adiacente, rîndurilor, rîndurilor adiacente se înţeleg defectele de alterare a informaţiei conţinute într-o celulă prin operaţii asupra unor celule dintr-o vecinătate a sa care este definită ca celulele adiacente, coloana ce conţine celula respectivă, coloanele adiacente, rîndul respectiv sau rîndurile adiacente. Aşa cum s-a menţionat anterior, noţiunea de adiacenţă nu implică neapărat o relaţie de adiacenţă topologică fizică.

Sensibilitatea la informaţia conţinută a fost analizată în capitolul 2.

Revenirea după un ciclu de scriere (Write Recovery) constă în capacitatea memoriilor de a efectua corect un ciclu de citire după unul sau mai multe cicluri de scriere. Actualele memorii semiconductoare de mare capacitate, utilizate în realizarea memoriilor operative nu mai prezintă de obicei acest tip de defect.

Regenerarea. Testarea regenerării constă în testarea capacităţii memoriei dinamice de a menţine informaţia un timp dat (2 ms de obicei) la o temperatură (de obicei 70°C) conform specificaţiei tehnice, fie a memoriei semiconductoare

controlate prin testare fie a unităţii de memorie realizate cu aceste memorii semiconductoare.

In principiu există două tipuri de teste pentru verificarea regenerării

1. In memorie se înscrie o configuraţie de date (considerată critică), apoi ea nu este activată un interval de timp dat, după care se verifică faptul că ea conţine aceeaşi informaţie.

2. In memorie se înscrie o configuraţie de date, apoi memoria este acţionată la frecvenţa de ciclu maximă, cu o asemenea secvenţă de adresare încît anumite rînduri să nu fie regenerate. După un interval de timp dat se verifică faptul că rîndurile care nu au fost regenerate nu au pierdut informaţia. Operaţia este repetată pînă cînd sînt testate toate rîndurile.

A doua metodă este utilizată cel mai des deoarece acţionarea memoriei duce la încălzirea pastilei de siliciu, simulîndu-se funcţionarea ei în condiţii reale. Trebuie avut în vedere că timpul de menţinere a informaţiei se garantează pentru temperatura ambiantă şi nu pentru temperatura pastilei de siliciu.

Aceste teste trebuie executate la o temperatură mai mare sau egală cu cea pentru care se garantează memoria.

Scrierile multiple pot fi datorate cuplajelor parazite dintre celule sau între linii ale decodificatoarelor de adresă, condiţionate de cuplaje cu liniile de scriere.

Revenirea amplificatoarelor de citire. După citirea unui şir lung de 0 sau 1, amplificatoarele tind să păstreze aceeaşi stare. Defectul se manifestă prin incapacitatea amplificatoarelor de a citi un 1 după citirea unui şir lung de zero-uri sau invers.

Pentru fiecare din defectele de mai sus au fost dezvoltate teste care le detectează cu o probabilitate mai mare sau mai mică.

Inainte de a trece la prezentarea acestora sînt necesare cîteva precizări de terminologie.

Prin <u>informaţie fundal</u> (background) se înţelege starea de 0 sau 1 în care sînt aduse toate celulele înainte de execuţia propriu-zisă a unui test. Vom nota informaţia fundal cu I.

Este posibilă o clasificare a secvenţelor de adresare ale testelor pentru memoriile semiconductoare. In afară de secvenţa scriere-citire, cel mai frecvent sînt utilizate sec-

venţele de tip MARCH, GALLOPING şi WALKING.

Secvenţa de tip MARCH se caracterizează prin faptul că pentru verificarea unei celule se execută o secvenţă tip citeşte I, scrie $\bar{I}$, citeşte $\bar{I}$, după care se trece la verificarea celulei următoare.

Secvenţa de tip GALLOPING, definită pe o mulţime N_i de celule care conţine şi celula C_i este caracterizată de faptul că pentru verificarea celulei C_i se înscrie în aceasta $\bar{I}$, apoi se citesc alternativ C_i şi C_j, $C_j \in N_i$. Dacă n este numărul de celule din N_i, atunci secvenţa pentru verificarea unei celuce conţine 2(n-1) citiri. Inainte de a trece la verificarea următoarei celule se reduce celula C_i în starea I.

Secvenţa de tip WALKING, definită pe o mulţime N_i de celule care conţine şi celula C_i, constă în trecerea celulei C_i în starea $\bar{I}$ şi apoi verificarea stării tuturor celulelor din N_i. Inainte de a trece la verificarea celulei următoare, celula C_i este redusă în starea iniţială.

Teste pentru verificarea memoriilor semiconductoare

In cele ce urmează vor fi tratate toate testele întîlnite în literatură. O sinteză a acestora, împreună cu timpii în care ele se pot executa este prezentată în tabelul A2.1o. Acest tabel a fost realizat prin completarea tabelului publicat de [FEL76] cu testele şi clasele de defecte propuse de alţi autori.

1. Testul checkerboard (tabla de şah) [MACR77].

Constă în înscrierea în memorie a unei informaţii tip tablă de şah şi apoi verificarea ei.

Număr de cicluri = 4 N.

Testul descoperă cu o probabilitate destul de mică anumite defecte de sensibilitate la informaţie.

2. Testul MARCH [MARC77].

Defecte detectate

++ neîndeplinirea condiţiei de unicitate a adresei

\+ perturbări ale celulei şi celulelor adiacente

+ perturbări în cadrul coloanelor şi al coloanelor adiacente.

Este testul cel mai răspîndit pentru verificarea unicităţii adresei.

Număr de cicluri: 10N.

Algoritm

1. Scrie 0 în celula i, i=o,1,...,N-1.

2. Citeşte I în celula i, scrie informaţia $\bar{I}$ în celula im i=o,1,...,N-1.

In urma acestui pas, memoria conţine informaţia fundal negată.

3. Repetă pasul 2 în ordine inversă, începînd cu celula N-1.

In urma acestui pas memoria conţine informaţia fundal iniţială.

4. Inscrie un fundal de "1" şi repetă paşii 2 şi 3.

Pentru scurtarea acestui test, fără reducerea esenţială a probabilităţii de detecţie a defectelor a fost dezvoltat testul MARCH II [FEL78].

Număr de cicluri 7N.

Algoritm

1. Inscrie informaţia fundal 0.

2. Verifică fiecare celulă printr-o secvenţă, citeşte I, scrie $\bar{I}$, citeşte $\bar{I}$.

In urma acestui pas, informaţia fundal este negată.

3. Repetă pasul 2 (cu informaţia fundal negată).

3. Testul MOVI [FEL78]

Este o secvenţă de test constînd din $\log_2(N)$ parcurgeri tip MARCH II ale memoriei, astfel încît fiecare rang de adresă să fie rangul cel mai puţin semnificativ al generatorului de adrese. $\log_2(N)$ este numărul de linii de adresă ale memoriei testate. Fiecare secvenţă de verificare tip MARCH II are 6N cicluri şi, împreună cu înscrierea informaţiei fundal, numărul total de cicluri este $N(6\log_2 N+1)$.

4. Testul GALCOL (coloană galopantă)

Caracteristica acestui test este secvenţa de verifica-

re tip Galloping pentru toate celulele dintr-un rînd.

Poate descoperi următoarele tipuri de defecte

++ viteză nesatisfăcătoare a decodificatorului de adrese coloane,

+ perturbări ale coloanei..

Prezentarea algoritmului se va face după [MACR77] deşi există mai multe variante ale acestui test.

Număr de cicluri $2(3N^{3/2}+6N)$.

Algoritm

Pentru a satisface necesităţile de regenerare adresa celulei testate va fi avansată cu 3, citindu-se 2 celule între etapele de testare. Exemplificarea se va face pentru memorii de 4K.

1. Se înscrie informaţia fundal "0".

2. Pentru a testa o celulă din coloana curentă se execută secvenţa

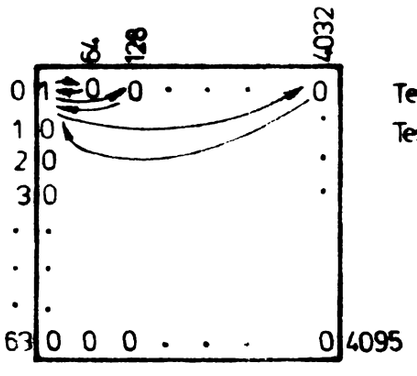
- se înscrie Ī în celula testată,
- se face o verificare tip GALLOPING pe celulele din rîndul respectiv.

Ex. testare coloana 0, celula 0

citeşte celula 0, citeşte celula 64

citeşte celula 0, citeşte celula 64

citeşte celula 0, citeşte celula 4032



A.2.1. Testul GALCOL

- citeşte celula testată, apoi înscrie I pentru a reface fundalul original,
- citeşte următoarele 2 celule pentru a regenera rîndurile respective.

Repetă secvenţa de testare pentru toate celulele din coloana curentă.

3. Repetă pasul 2 pentru toate coloanele.

4. Se înscrie informaţia fundal "1" şi se repetă paşii 2 şi 3.

5. Testul GALROW (Rînd galopant)

Acest test se aseamănă cu cel anterior, cu deosebirea că secvenţa de verificare tip galloping se execută pe mulţimea celulelor dintr-o coloană. Datorită faptului că se execută citiri în fiecare rînd la intervale scurte, nu necesită regenerare dar nici nu poate detecta acest tip de defecte.

Poate detecta următoarele tipuri de defecte

++ viteză nesatisfăcătoare a decodificatorului de adresă de rînd (selecţia rîndurilor)

++ perturbări ale rîndului

\+ perturbări ale celulelor şi rîndurilor adiacente prin citire.

Număr de cicluri $4(N^{3/2}+2N)$

Algoritm

1. Se înscrie informaţia de fundal 0 în toate celulele.

2. Pentru a testa o celulă din rîndul curent se execută secvenţa

- se înscrie Ī în celula testată
- se face o verificare tip GALLOPING pe celulele din coloana respectivă

Ex. testare rînd 0, celula 0

citeşte celula 0, citeşte celula 1

citeşte celula 0, citeşte celula 2

.

.

.

citeşte celula 0, citeşte celula 63

- se citeşte din nou celula testată, apoi se înscrie I pentru a reface fundalul original
- repetă secvenţa de verificare pentru toate celulele din rîndul curent.

3. Repetă pasul 2 pentru toate rîndurile.

4. Se înscrie informaţia de fundal 1 şi se repetă paşii 2 şi 3.

6. Testul GALDIA (Galloping pe diagonală)

Acest test poate detecta următoarele tipuri de defecte

++ unicitatea adresei

++ viteza decodificatoarelor de adresă

+ perturbări ale celulei.

Pentru testul GALDIA se partiţionează o memorie cu N celule în $\sqrt{N}$ submulţimi, (diagonale, fiecare conţinînd $\sqrt{N}$ celule). Fie x şi y coordonatele unei celule, astfel încît adresa sa e dată de relaţia

$$adr = x + \sqrt{N} \cdot y \qquad (A2.1)$$

operaţia efectuîndu-se în mulţimea claselor de resturi modulo N.

Considerăm că operaţiile de adunare efectuate asupra valorilor coordonatelor se efectuează în mulţimea claselor de resturi modulo $\sqrt{N}$.

$$B = x_B + \sqrt{N}\, y_B \qquad (A2.2)$$

Fie adresa unei celule. Adresele celorlalte celule din aceeaşi diagonală cu pantă pozitivă sînt date de relaţia

$$A(B,n) = \left[(x_B - n) \bmod \sqrt{N} + \sqrt{N}(y_B + n) \bmod \sqrt{N}\right] \bmod N \qquad (A2.3)$$

$n = 0, , \ldots \sqrt{N}-1$

Pentru o diagonală cu pantă negativă, relaţia este

$$A(B,N) = \left[(x_B + n) \bmod \sqrt{N} + \sqrt{N}(y_B + n) \bmod \sqrt{N}\right] \bmod N \qquad (A2.4)$$

$n = 0,1,\ldots, \sqrt{N}-1$

De exemplu fie o memorie cu N=64.

0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

y

x

A.2.2. Memoria M 64

Se definesc următoarele 8 diagonale cu pantă pozitivă, Tabelul A2.1. Diagonalele cu pantă pozitivă din M64.

0	15	22	29	36	43	5o	57
1	8	23	3o	37	44	51	58
2	9	16	31	38	45	52	59
3	1o	17	24	39	46	53	6o
4	11	18	25	32	47	54	61
5	12	19	26	33	4o	55	62
6	13	2o	27	34	41	48	63
7	14	21	28	35	42	49	56

şi următoarele 8 diagonale cu pantă negativă.

Tabelul A2.2. Diagonalele cu pantă negativă din M64.

0	9	18	27	36	45	54	63
1	1o	19	28	37	46	55	56
2	11	2o	29	38	47	48	57
3	12	21	3o	39	4o	49	58
4	13	22	31	32	41	5o	59
5	14	23	24	33	42	51	6o
6	15	16	25	34	43	52	61
7	8	k7	26	35	44	53	62

Esenţa testului de galloping pe diagonală este o secvenţă de verificare tip galloping pe mulţimea celulelor dintr-o diagonală definită conform relaţiei (A2.3) sau (A2.4).

Număr de cicluri $4(N^{3/2}+2N)$

Algoritm [MACR77 corectat]

1. Se înscrie informaţia de fundal "0"
2. Pentru a testa o celulă se execută
   - se înscrie Ī în celula testată
   - se face o verificare tip galloping pe diagonala respectivă

Ex. Verificarea celului 8 din M64
citeşte 0, citeşte 23 (n=1)
citeşte 8, citeşte 3o (n=2)

citeşte 8, citeşte 37 (n=3)
•
•
•
citeşte 8, citeşte 1 (n=7)

- se citeşte din nou celula testată apoi se înscrie I în ea, refăcînd fundalul iniţial,
- se repetă secvenţa de mai sus pentru toate celulele.

3. Se înscrie informaţia de fundal "1" şi se repetă pasul 2.

7. Testul WALKPAT [MONT75], [MACR77]

Acest test poate detecta următoarele tipuri de defecte
++ unicitatea adresei
+ perturbări în cadrul coloanelor
+ perturbări în coloanele adiacente.

Număr de cicluri $2(N^2+2N)$.

După cum se va observa din algoritm acest test, descoperă cu probabilitate maximă defecte de selecţie multiplă sau celule nefuncţionale (unicitatea adresei). Performanţele sale nu depăşesc însă cu mult testul MARCH iar durata sa îl face prohibitiv pentru memorii cu capacitate mare. Neîndeplinirea condiţiei de unicitate a adresei este detectată cu aceeaşi probabilitate şi de testul GALPAT care în plus scoate în evidenţă şi alte tipuri de defecte. Din acest motiv este puţin utilizat.

Algoritm

1. Se înscrie informaţia de fundal "0"
2. Se înscrie $\bar{I}$ în celula C_i şi se citeşte starea tuturor celulelor.

Se înscrie I în celula C_i pentru a reface fundalul iniţial.

Se repetă secvenţa de mai sus pentru i=o,1,...,N-1.

3. Se înscrie informaţia de fundal "1", şi se repetă pasul 2.

8. Testul GALPAT [TEKT74], [MONT75], [MACR77]

Esenţa acestui test este că pentru verificarea unei celule se execută o secvenţă de verificare tip GALLOPING pe toa-

te celulele memoriei.

Defecte detectate

++ unicitatea adresei

++ viteza decodificatoarelor de adresă

++ perturbări în cadrul coloanelor

\+ perturbări ale celulelor

\+ perturbări ale celulelor adiacente

\+ perturbări în cadrul coloanelor adiacente

Număr de cicluri $2(2N^2+N)$

Algoritm

1. Se înscrie informaţia de fundal "0"

2. Pentru verificarea celului C_i se execută secvenţa

- înscrie $\bar{I}$ în celula C_i

- se face o verificare tip Galloping pe toată memoria

Ex. Verificarea celului 22

citeşte celula 22, citeşte celula 23

citeşte celula 22, citeşte celula 24

.....

citeşte celula 22, citeşte celula N

citeşte celula 22, citeşte celula 0

citeşte celula 22, citeşte celula 1

.....

citeşte celula 22, citeşte celula 21

- înscrie I în C_i pentru a reface fundalul iniţial.

Repetă secvenţa de mai sus pentru toate celulele.

3. Se înscrie informaţia fundal "1" şi se repetă pasul 2.

9. Testul MASEST [MACR77]

Defecte detectate : nefuncţionarea decodificatoarelor de adresă.

Număr de cicluri 10N

Algoritm

1. Se înscrie o informaţie fundal de 0 şi 1 alternativ.

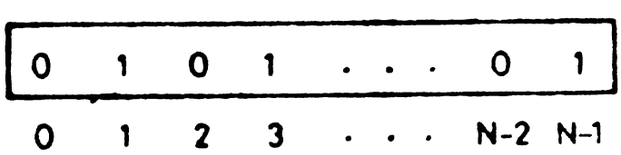


Fig.A2.3.

2. Testează cele N celule, cu următoarea secvenţă pentru celula C_i.

Citeşte C_i, citeşte C_{N-1-i}, citeşte C_i

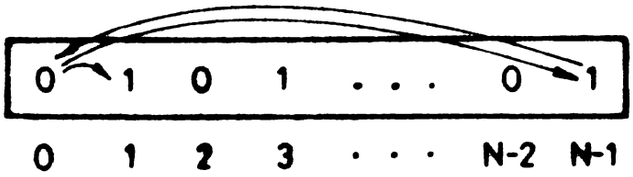


Fig.A2.4.

3. Citeşte toate celulele secvenţial.

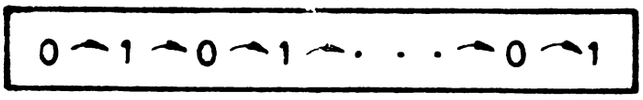


Fig.A2.5.

4. Inscrie informaţia de fundal complementară şi repetă paşii 2 şi 3.

lo. Testul DIAPAT [MACR77]

Acest test a fost special conceput pentru testarea reveririi amplificatoarelor de citire.

Alţi autori denumesc acest test "SHIFTED DIAGONAL" [TH.T74] sau "MOVING DIAGONAL" [OWEN79a].

Spre deosebire de testul GALDIA unde fiecare diagonală era formată din $\sqrt{N}$ celule, pentru acest test diagonala este definită ca avînd între 1 şi $\sqrt{N}$ celule, existînd $2\sqrt{N}-1$ diagonale.

In fig.A2.6 sînt reprezentate diagonalele cu pantă pozitivă din M64.

Pentru diagonalele cu pantă pozitivă, care încep în celule din coloana 0, adresele celulelor din diagonală sînt date de relaţia

$$A = B+(\sqrt{N}-1)n, \quad n=0,1,2,\ldots,\sqrt{N}-1 \qquad (A2.5)$$

Adresele de început ale diagonalelor care încep în rîndul $\sqrt{N}-1$, mai puţin cea care începe în celula $\sqrt{N}-1$ sînt date de relaţia

$$B = 2\sqrt{N}-1 + \sqrt{N}\cdot K \qquad (A2.6)$$

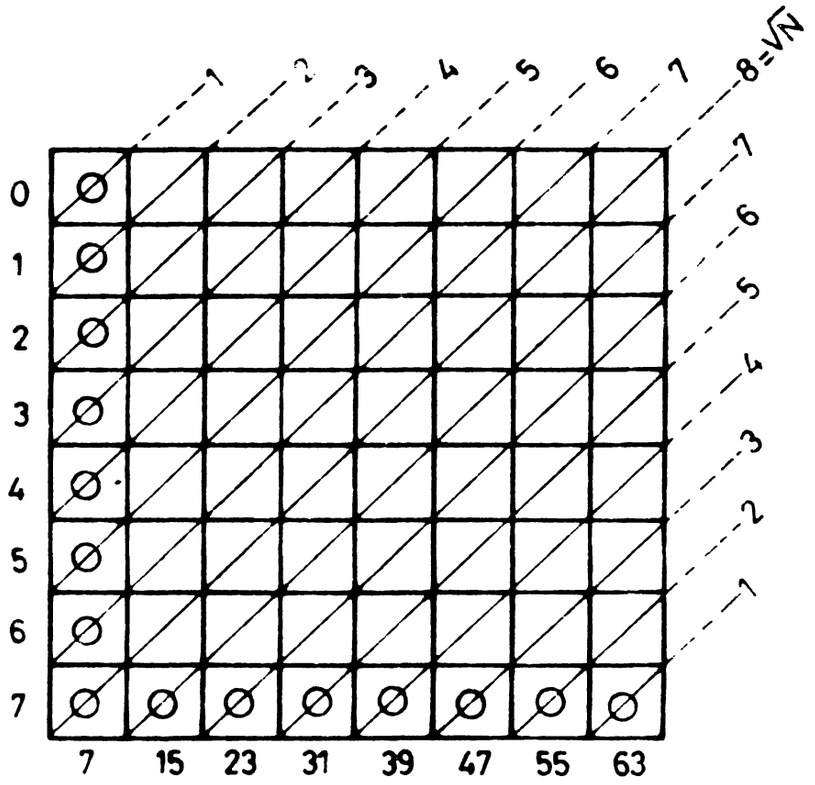


Fig.A2.6. Diagonalele cu pantă pozitivă din M64. Celulele marcate cu cercuri sînt celule de început ale diagonalelor

Adresele celulelor din aceste diagonale sînt date de relaţia

$$A(n) = B+(\sqrt{N}-1)n, \quad n=0,1,2,\ldots,(\sqrt{N}-2-k) \qquad (A2.7)$$

In tabelul de mai jos sînt date diagonalele pozitive ale memoriei M64.

Tabelul A2.3 Diagonalele cu pantă pozitivă din M64.

0
1 8
2 9 16
3 10 17 24
4 11 18 25 32
5 12 19 26 33 40
6 13 20 27 34 41 48
7 14 21 28 35 42 49 56
15 22 29 36 43 50 57
23 30 37 44 51 58
31 38 45 52 59
39 46 53 60
47 54 61
55 62
63

Număr de cicluri : $2(2N^{3/2}+2N)$.

[MACR77]indică în mod eronat numărul de cicluri pentru acest test ca fiind $2(2N^{3/2}+5N^{1/2}+4N)$.

Algoritm pentru diagonale cu pantă pozitivă.

•Pasul 1. Se înscrie o informaţie fundal "o" în toate celulele.

•Pasul 2.

- se înscrie $\bar{I}$ în diagonala curentă D_i
- se citesc toate celelalte diagonale (I)

  Adresa diagonalei următoare se obţine adunînd $\sqrt{N}$-1 la adresa ultimului element din diagonala curentă.

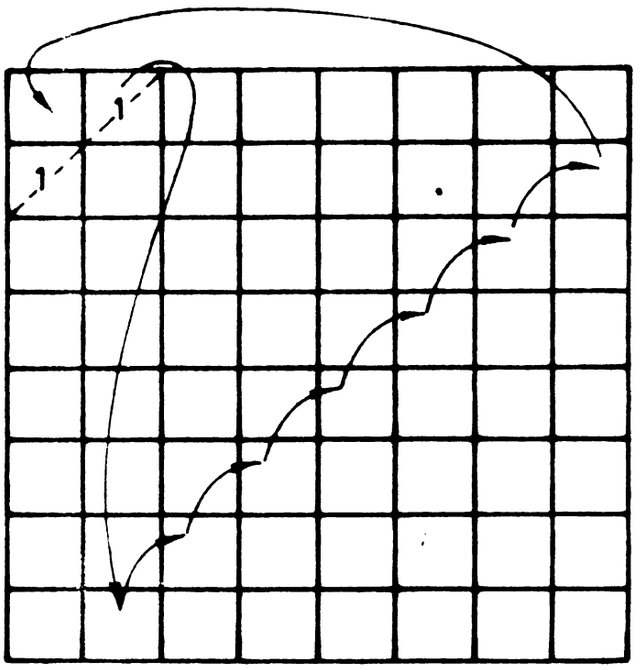


Fig.A2.7. Testul DIAPAT - secv.de adresare

- se citeşte diagonala curentă $\bar{I}$
- se înscrie I în diagonala curentă pentru a reface informaţia fundal iniţială.

Secvenţa de mai sus se repetă pentru fiecare diagonală.

•Pasul 3. Se înscrie informaţia fundal "1" în toate celulele şi se repetă pasul 2.

11. Testul WAKCOL (Walking column) [MACR77]

Tipuri de defecte detectate:

++ perturbări în cadrul coloanei

++ revenire lentă a amplificatoarelor de citire.

Numărul de cicluri $2(N^{3/2}+3N)$.

Algoritm

- Pasul 1. Se înscrie o informaţie fundal "0".
- Pasul 2.
  - se înscrie $\bar{I}$ în toate celulele unei coloane
  - se citesc secvenţial celelalte celule

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

Fig.A2.8. Testul WAKCOL - secvenţă de adresare

  - se citesc celulele din coloana curentă
  - se înscrie I în toate celulele coloanei curente pentru a reface informaţia fundal iniţială.

Se repetă secvenţa de mai sus pentru toate coloanele.

- Pasul 3. Se înscrie informaţia fundal "1" şi se repetă pasul 2.

12. Testul DUAL WAKOL [MACR77]

Tipuri de defecte detectate

++ perturbări în cadrul coloanelor

++ revenire lentă a amplificatoarelor de citire.

Nr.de cicluri $2\left[(1/2)N^{3/2}+3N\right]$.

Algoritm

Pasul 1. Se înscrie informaţia fundal "0".

Pasul 2.

Fie j indicele curent al coloanelor testate la o trecere.

Se vor testa coloanele j şi $j+\frac{1}{2}\cdot\sqrt{N}$, $j=0,1,\ldots,\frac{1}{2}\cdot\sqrt{N}-1$

- Se înscrie $\bar{I}$ în coloanele j şi $j+\frac{1}{2}\cdot\sqrt{N}$ (fig.A2.9).

- Se citesc secvenţial toate celulele memoriei, începînd cu coloana $j+\frac{1}{2}\cdot\sqrt{N}+1$, (celula $\sqrt{N}(j+\frac{1}{2}\cdot\sqrt{N}+1)$ (fig.A2.10)

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

Fig.A2.9. DUAL WAKOL Informaţia conţinută

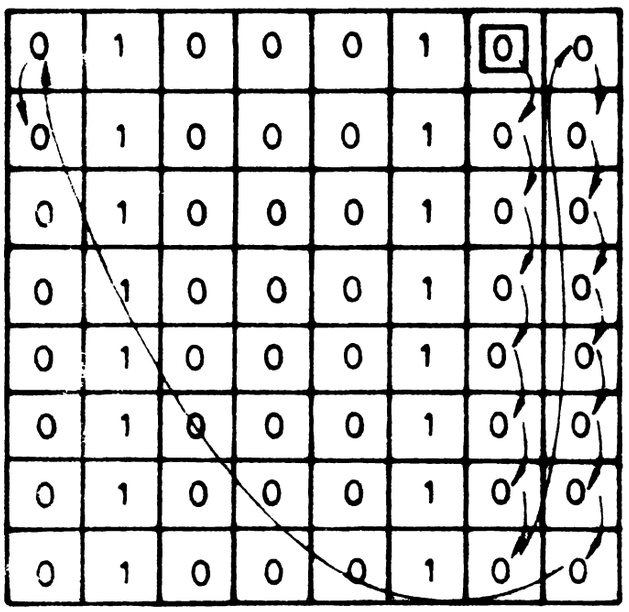


Fig.A2.1o. DUAL WAKOL - Secvenţă de adresare

- Se înscrie I în coloanele j şi $j+\frac{1}{2}\cdot\sqrt{N}$ pentru a reface fundalul original.

Se repetă secvenţa de mai sus pentru toate perechile de coloane.

Pasul 3. Se înscrie informaţia fundal "1" şi se repetă pasul 2.

13. Testul HAFCOL [MACR77]

Acest test este similar cu testul WAKOL însă informaţia conţinută în jumătatea inferioară a fiecărei coloane este complementul celei conţinute în jumătatea superioară. In acest fel se ţine seama de faptul că majoritatea memoriilor dinamice, înscriind o informaţie fundal identică în toate celulele, celulele din jumătatea inferioară vor fi în starea negată.

Tipuri de defecte detectate

++ perturbări în cadrul coloanelor

++ revenire lentă a amplificatoarelor de citire

Număr de cicluri $2(N^{3/2}+3N)$.

Algoritm

•Pasul 1. Se înscrie o informaţie fundal care este "0" în fiecare celulă din jumătatea superioară a matricii şi "1" în cea inferioară a (fig.A2.11).

•Pasul 2.

Pentru fiecare coloană se execută secvenţa.

- se înscrie în coloană curentă complementul stării aşteptate a acelei coloane, $\bar{I}$

- se citesc secvenţial toate celulele începând din coloana ce urmează după coloana curentă, inclusiv celulele din coloana curentă (fig.A2.12).

- se înscrie în coloana curentă informaţia iniţială de fundal.

•Pasul 3.

Se înscrie în toată memoria informaţia de fundal complementară celei înscrise în pasul 1 şi se repetă pasul 2.

14. Testul HAMPAT [MACR77]

Acest test detectează defecte de tipul scrierilor multiple în cadrul aceleiaşi coloane.

Număr de cicluri $2(N^{3/2}+2N)$.

Algoritm

•Pasul 1. Se înscrie o informaţie fundal "0" în toată memoria.

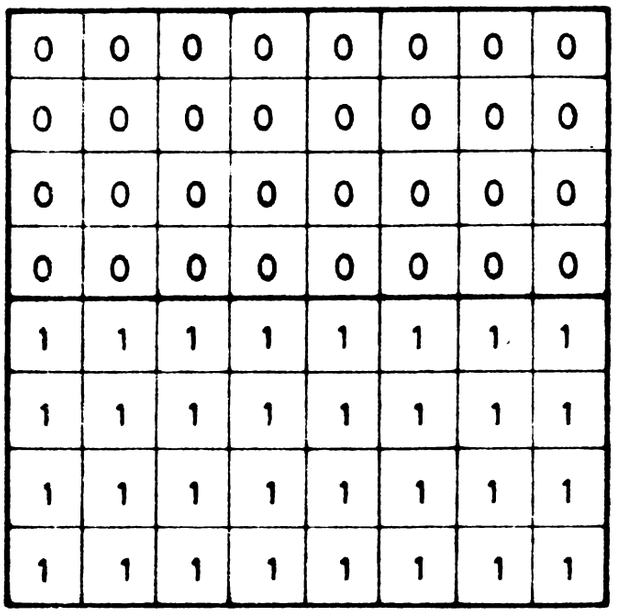


Fig.A2.11. HAFCOL - Informaţia fundal

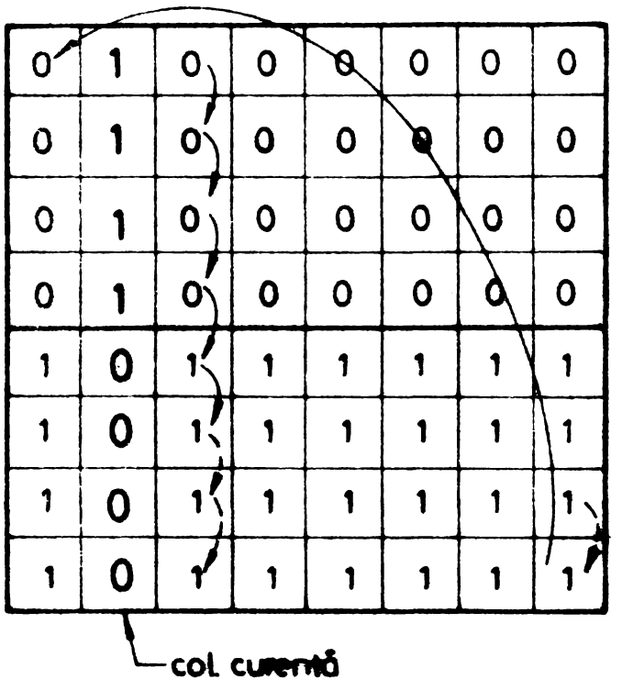


Fig.A2.12. HAFCOL - Secvenţă de adresare

- Pasul 2.

  Pentru fiecare coloană se repetă secvenţa:

  Pentru fiecare celulă din coloana curentă se repetă secvenţa;

- scrie $\bar{I}$, scrie I, scrie $\bar{I}$ în celula curentă

- se citesc toate celulele coloanei curente începînd cu celula de după celula curentă (fig.A2.13) şi inclusiv celula curentă,

- se înscrie I în celula curentă pentru a restabili informaţia fundal.

• Pasul 3.

Se înscrie informaţia fundal "1" şi se repetă pasul 2.

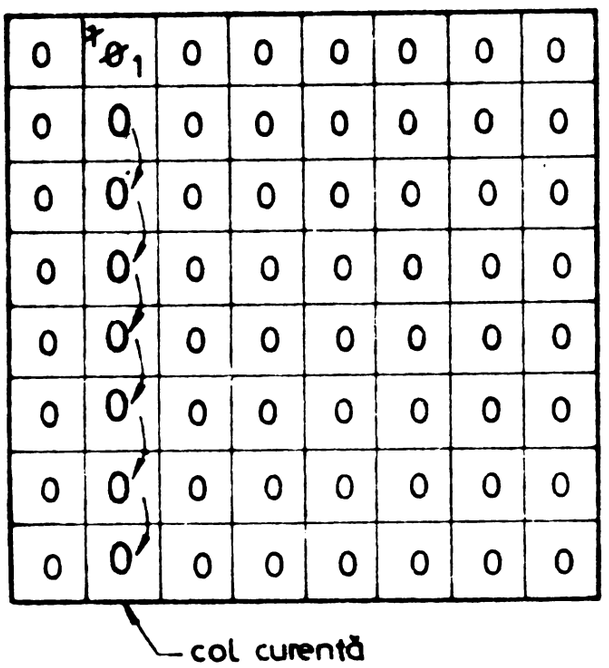


Fig.A2.13. HAMPAT - Secvenţă de adresare

15. Testul DISTURBED WALKING ONE/ZERO [MONT75]

Acest test verifică faptul că fiecare celulă nu este perturbată de o operaţie de scriere/citire în celulele adiacente. Denumirea provine de la faptul că memoria conţine un singur "1" care se deplasează, fiind perturbat la fiecare poziţie.

Număr de cicluri : $2N(4+8K)$, K fiind numărul de activări a fiecărei celule adiacente.

Algoritm

Pasul 1.

- se înscrie informaţia fundal "0" în toate celulele.

Pasul 2.

- se înscrie în celula curentă $\bar{I}$.

- se efectuează K operaţii de citire şi/sau scriere în

fiecare din celulele adiacente (fig.A2.14).

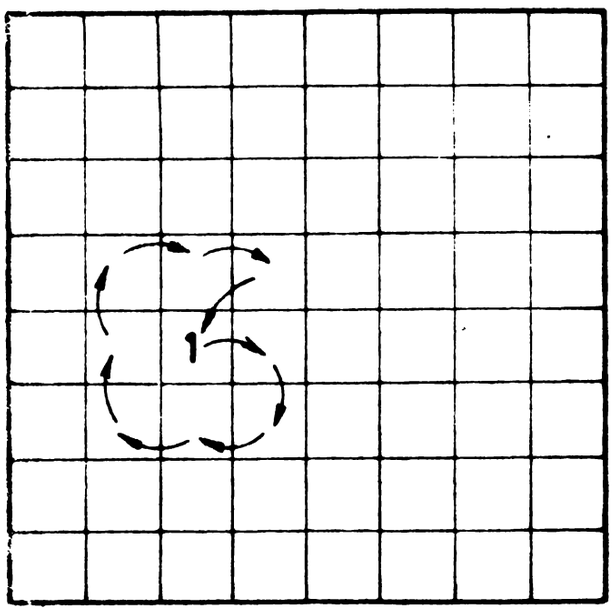


Fig.A2.14. DISTURBED WALKING ONE/ZERO - Secvenţă de adresare

- se citeşte celula curentă.
- se înscrie I în celula curentă pentru a reface informaţia fundal iniţială.

Se repetă secvenţa de mai sus pentru toate celulele.

- Pasul 3.
- se înscrie informaţia fundal "1" şi se repetă pasul 2.

16. <u>Testul X-Y complementat</u> [MONT75]

[FEE78] denumeşte acest test ADDRESS COMPLIMENT, test de adresă complementată.

Defecte detectate:

+ viteză necorespunzătoare a decodificatoarelor de adresă

Număr de cicluri : 4N.

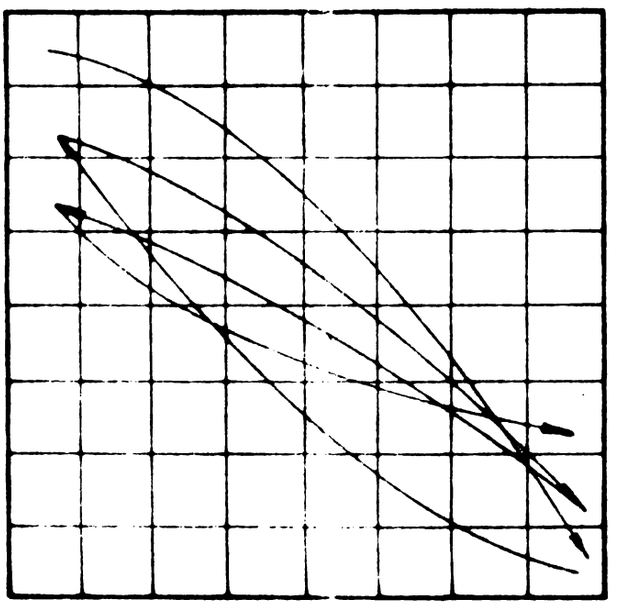
Caracteristic acestui test este secvenţa de adresare care se construieşte în aşa fel încît de la un ciclu la altul un număr maxim de linii de adresă să treacă în stare complementară

Adresa poate fi generată de un numărător, care numără de la 0 la N/2-1. A fiind valoarea curentă a numărătorului, se efectuează operaţii în celula A şi $\bar{A}$.

Tabelul A2.4. Secvenţa de adresare X-Y complementat.

Numărul operaţiei	Adresa la care se efectuează	A
0	0 0 0 0 0 0 0 0 0	0
1	1 1 1 1 1 1 1 1 1	
2	0 0 0 0 0 0 0 0 1	1
3	1 1 1 1 1 1 1 1 0	
4	0 0 0 0 0 0 0 1 0	2
5	1 1 1 1 1 1 1 0 1	
- - - -	- - - - - - - - - -	---
62	0 0 0 1 1 1 1 1 1	31
63	1 1 1 0 0 0 0 0 0	

In fig.A2.15a este prezentată secvenţa de adresare iar în fig.A2.15b în fiecare celulă din M64 s-a înscris numărul operaţiei de scriere sau citire efectuată asupra acelei celule.



0	16	32	48	63	47	31	15
2	18	34	50	61	45	29	13
4	20	36	52	59	43	27	11
6	22	38	54	57	41	25	9
8	24	40	56	55	39	23	7
10	26	42	58	53	37	21	5
12	28	44	60	51	35	19	3
14	30	46	62	49	33	17	1

a. b.

Fig.A2.15. X-Y Complementat - secvenţă de adresare

Algoritm

- se înscrie o informaţie tablă de şah sau paritatea adresei,
- se citeşte această informaţie
- se înscrie informaţia complementată
- se citeşte această informaţie.

Obs: [FEE78] defineşte secvenţa de adresare cu adresa complementată ca fiind o secvenţă în care de la un ciclu la altul toţi biţii de adresă în afară de unul basculează în starea complementară.

17. Testul GWR (Galloping Write Recovery) [MONT75], [FEE73]

Defecte detectate:

++ unicitatea adresei

++ viteză nesatisfăcătoare a decodificatoarelor de adresă

++ perturbări în cadrul coloanei

++ revenire după ciclu 1 de înscriere

Număr de cicluri $2(2N^2+N)$

Algoritmul este identic ca secvenţă de adresare cu GALPAT deosebirea fiind că pentru testarea celulei curente se fac alternativ o citire a celulei curente şi o scriere a altei celule.

Ex. Verificarea celulei 22

citeşte $\bar{I}$ în celula 22, scrie I în celula 23

citeşte $\bar{I}$ în celula 22, scrie I în celula 24

.....

citeşte $\bar{I}$ în celula 22, scrie I în celula N-1

citeşte $\bar{I}$ în celula 22, scrie I în celula 0

.....

citeşte $\bar{I}$ în celula 22, scrie I în celula 21

18. Testul Paritatea adresei [MONT75]

Defecte detectate

+ sensibilitate la informaţia înscrisă

+ celule blocate pe 1 sau 0

+ circuitele de decodificare pe rînd şi coloană.

Număr de cicluri 4N.

Algoritm

•Pasul 1. Se înscrie succesiv în fiecare celulă un bit de paritate pasă al adresei celulei respective.

•Pasul 2. Se citesc succesiv toate celulele.

•Pasul 3. Se înscrie paritatea impară în toate celulele.

•Pasul 3. Se citesc şi se verifică succesiv toate celulele.

Obs. Fie matricea elementară

$$E = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Dacă memoria are un număr par de linii de adresă matricea informaţiei conţine în memorie este de forma

$$I = \begin{bmatrix} E & \bar{E} & E & \dots & \bar{E} \\ \bar{E} & E & \bar{E} & \dots & E \\ E & \bar{E} & E & \dots & \bar{E} \\ \vdots & & & & \\ \bar{E} & E & \bar{E} & \dots & E \end{bmatrix}$$

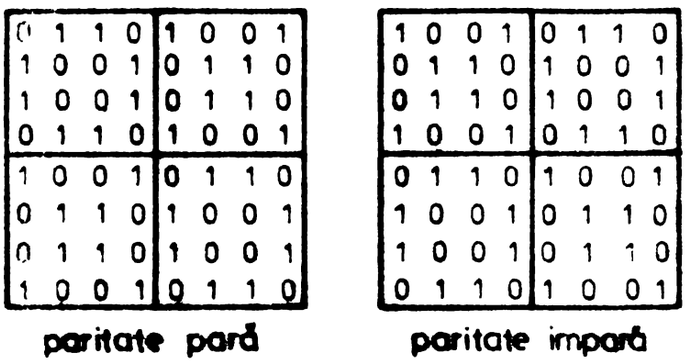


Fig.A2.16. Paritatea adresei

Acest test poate detecta doar acele defecte ale decodificatoarelor la care adresa celulei activate intern are paritatea diferită de cea a celulei adresate.

19. Testul CHEKCOL [MACR77]

Este un test tip tablă de şah care se deplasează pe coloane.

Tipuri de defecte detectate

+ perturbări în cadrul coloanei

+ selecţii interne multiple

Număr de cicluri $2(\frac{1}{4} N^{3/2} + \frac{1}{4} N^2 + 2N)$

Algoritm

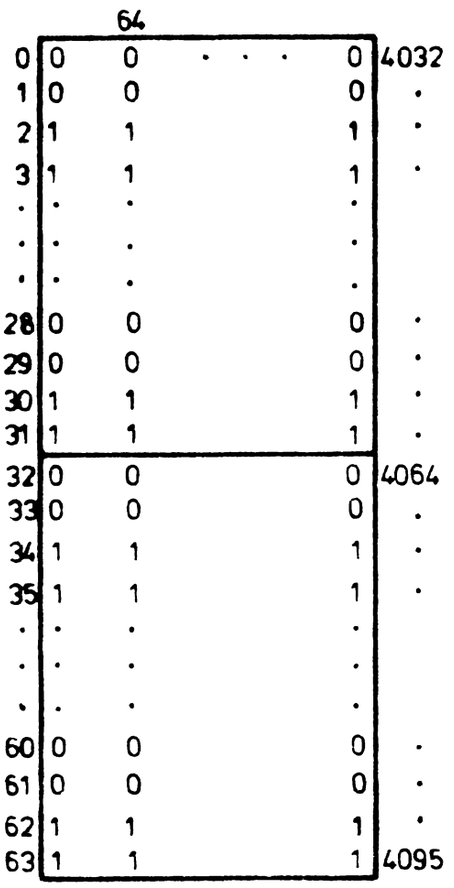
•Pasul 1.

Se înscrie o informaţie fundal constînd din perechi de

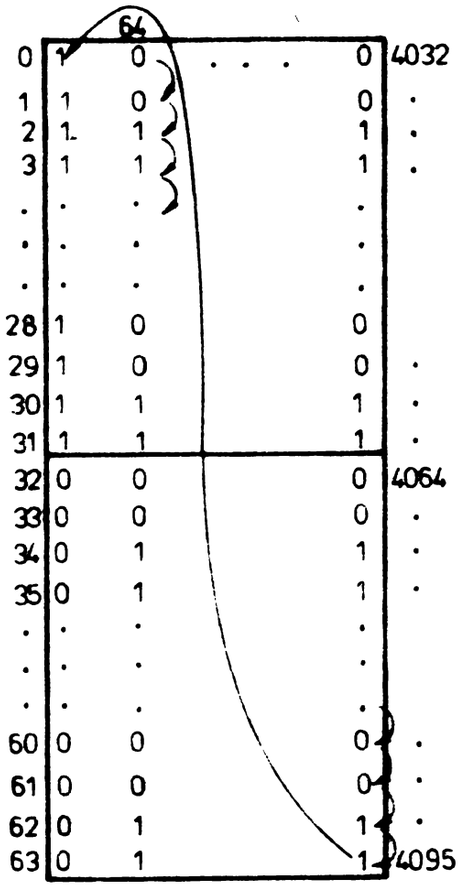
şi perechi de 1 alternativ (fig.A2.17).

• Pasul 2.

- se modifică conţinutul unui bloc de celule de lungime egală cu cea a unei coloane, înscriind "0" în primele $\sqrt{N}/2$ celule şi "1" în următoarele $\sqrt{N}/2$ (fig.A2.18)



A.2.17. CHEKOL - Inf. fundal (M4096)



A.2.18. CHEKOL - pasul 2 (M4096)

- se citesc toate celulele începînd cu celula ce urmează după acest bloc, inclusiv celulele din blocul de lungine $\sqrt{N}$.

- se reface informaţia iniţială din primele 4 celule ale blocului (fig.A2.19).

• Pasul 3.

Prima celulă a blocului va fi acum celula 4. Se repetă procedura de la pasul 2 depăşind de fiecare dată blocul cu 4 celule.

• Pasul 4.

- Se înscrie informaţia fundal complementară, adică 11001100 ... 1100 şi se repetă paşii 2 şi 3.

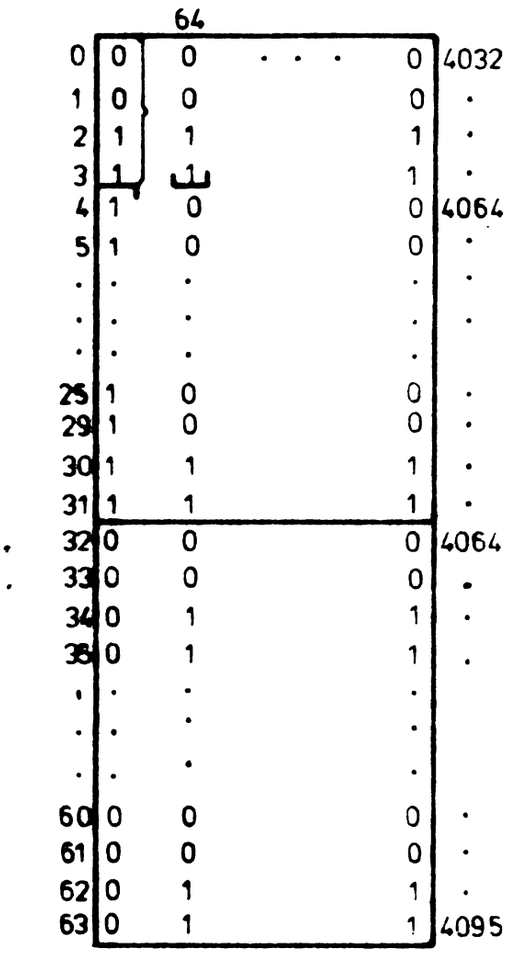


Fig.A2.19. CHEKCOL - Deplasarea blocului în poziţiile 4-67

2o. <u>Testul Knaizuk</u> [KNAI77a],[KNAI77b]

Acest test a fost definit în anul 1977.

Tipuri de defecte detectate:

+ defecte ale registrelor de adresă şi de decodoarelor de adresă
+ defecte ale matricii de memorare (celule nefuncţionale)

Număr de cicluri: 4N.

Fie A_j adresa binară a celulei care are adresa j, j=0, 1,...,N-1.

Se partiţionează memoria în 3 submulţimi

$$\begin{aligned} \pi_0 &= \{ A_\mu \mid \mu = 0 \text{ (modulo 3)} \} \\ \pi_1 &= \{ A_\mu \mid \mu = 1 \text{ (modulo 3)} \} \\ \pi_2 &= \{ A_\mu \mid \mu = 2 \text{ (modulo 3)} \} \end{aligned} \qquad (A2.5)$$

Algoritm

•Pasul 1.

Se înscrie "0" în toate celulele ale căror adrese îndeplinesc relaţia

$A_j \in \pi_1$ şi $A_k \in \pi_2$

•Pasul 2.

Se înscrie "1" în toate celulele cu adrese din π_o, $A_i \in \pi_o$

•Pasul 3.

Se citesc toate celulele cu $A_j \in \pi_1$.

•Pasul 4.

Se înscrie "1" în toate celulele cu $A_j \in \pi_1$.

•Pasul 5.

Se citesc toate celulele cu $A_k \in \pi_2$, al căror conţinul trebuie să fie "0".

•Pasul 6.

Se citesc toate celulele cu $A_i \in \pi_o$ şi $A_j \in \pi_1$, al căror conţinut trebuie să fie "1".

•Pasul 7.

Se înscrie "0" în toate celulele cu $A_i \in \pi_o$ şi apoi se citesc aceste celule.

•Pasul 8.

Se înserie "0" în toate celulele cu $A_k \in \pi_2$ şi apoi se citesc aceste celule.

Sub formă de tabel acest algoritm poate fi reprezentat astfel.

Tabelul A2.5.

Pasul / Partiţia	1	2	3	4	5	6	7	8
π_o		W"1"				R"L"	W"0",R"0"	
π_1	W"0"		R"0"	W"1"		R"1"		
π_2	W"0"				R"0"			W"1",R"1"

21. Testul Knaizuk-Nair [NAIR79]

R.Nair [NAIR79], a făcut în 1979 o perfecţionare a testului de mai sus, astfel:

Modificarea nr.1. Proprietăţile testului nu se modifică dacă scrierea din pasul 7 este inclusă în pasul 1 iar citirea este intercalată imediat după pasul 1.

Modificarea nr.2. Pasul 8 este intercalat după pasul 5.

Astfel modificată, reprezentarea tabelară a algoritmului este cea din tabelul A2.6.

Tabelul A2.6.

Pasul / Partiţie	1	2	3	4	5	6	7	8
π_0	W"0"	R"0"	W"1"					R"1"
π_1	W"0"			R"0"	W"1"			R"1"
π_2	W"0"					R"0"	W"1"	R"1"

[NAIR79] demonstrează că testul este validat pentru orice partiţionare π^k de forma

$$\pi_i^k = \{A_\mu \mid \mu \equiv i \text{ (modulo } k)\} \quad i=0,1,\ldots,k-1 \qquad (A2.9)$$
$$k \neq 2^j,\ j < \lceil \log N \rceil$$

Interesant este cazul k=N, caz în care testul ia forma testului MARCH (tabelul A2.7).

Tabelul A2.7.

Pas / Adr.	1	2	..	N-1	N	N+1	N+2	N+3	N+4	..	3N-3	3N-2	3N-1	3N
0	W0					R0	W1							
1		W0						R0	W1					
...			..							..				
N-2				W0							R0	W1		
N-1					W0								R0	W1

3N+1	3N+2	...	4N+1	4N
R1				
	R1			
		...	R1	
			R1	
				R1

[NAIR79] demonstrează capacitatea acestui test de a descoperi defecte de neîndeplinire a condiţiei de unicitate. a adresei.

22. Testul Milnor - MILPAT [MILN8o]

Acest test, foarte recent, a fost dezvoltat de specialişti ai firmei National Semiconductor pentru testarea C.I. de memorie dinamice. Pentru aplicarea sa nu este necesară adresarea topologică sau secvenţe speciale de adresare.

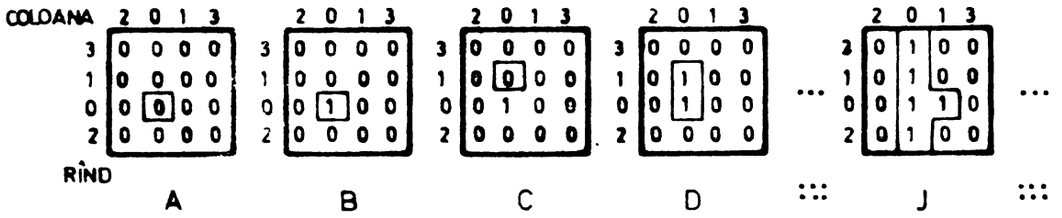
Defecte detectate :

++ cuplaje parazite între celule

++ pierderea informaţiei la memorii dinamice în prezenţa perturbaţiilor

++ neîndeplinirea condiţiei de unicitate a adresei.

In fig.A2.2o sînt indicate modificările suferite de o memorie cu 16 celule, cu o topologie dată, în cursul acestui test.



A.2.20. Testul MILPAT

Algoritm

- Pas 1. Se înscrie o informaţie fundal "0".
- Pas 2.
  - Se citeşte celula curentă
  - Se scrie Ī în celula curentă
  - Se citeşte celula curentă de X ori, unde

$$X = \frac{\text{Timp între regenerări}}{(N_R-1)\times T_C} - 2 \qquad (A2.1o)$$

unde N_R - numărul de rînduri

T_C - durata ciclului memoriei.

Se repetă ciclul de mai sus pentru toate celulele, adresarea făcîndu-se pe coloane.

•Pas 3. Se repetă pasul 2, în memorie fiind acum o informaţie fundal "1".

După cum se poate observa este o secvenţă tip MARCH modificată.

Durata testului:

$$T_T = N.T_C + 2 \cdot \frac{N}{N_R-1} \cdot T_R \qquad (A2.11)$$

unde N - numărul total de celule

N_R - numărul de rînduri

T_C - ciclul memoriei, la care se efectuează testul

T_R - timpul admis între două regenerări ale aceleiaşi celule.

Cu T_R=4 ns şi T_C=5oo ns, durata testului pentru un C.I. de 16K este de 1,o4 s.

23. Testul NAIR-A [NAIR78]

Acest test, elaborat de R.Nair satisface condiţiile 1, 2 şi 3 prezentate la capitolul 2.2.4.

Defecte detectate

++ neîndeplinirea condiţiei de unicitate a adresei

++ cuplaje între oricare 2 celule

Număr de cicluri 3oN-8.

Algoritm

Algoritmul, prezentat în tabelul A2.8, este format din 8 secvenţe care parcurg memoria atît în sens crescător cît şi în sens descrescător şi două secvenţe de iniţializare.

24. Testul NAIR-B [NAIR78]

Defecte detectate

++ absenţa uneia sau a mai multor celule

++ cuplaje între 3 celule

+ viteză necorespunzătoare a decodificatoarelor de adresă.

Număr de cicluri $N(1+32 \log_2 N)$.

Algoritm

Acest test conţine un set de secvenţe ce se repetă de un număr de ori egal cu cel al liniilor de adresă. De fiec-

Tabelul A2.8. Algoritmul de test NAIR-A

Adr. cel.	Init.	Secvenţa 1	Secvenţa 2	Secvenţa 3	Secvenţa 4
0	0	R↑ R	R↑ R	R↑	R↑
1	0	R↑ R	R↑ R	R↑ R	R↑ R
2	0	R↑ R	R↑ R	R↑ R	R↑ R
.	.	.. .	.. .	R↑ R	.. .
N-2	0	R↑ R	R↑ R	R↑ .. . R	R↑ R
N-1	0	R↑	R↑	R↑ R	R↑ R

Adr. cel.	Secvenţa 5	Secvenţa 6	Reset	Secvenţa 7	Secvenţa 8
0	R↑↓ R	R↑↓	1	R↑↑ R	R↑↑
1	R↑↓ R	R↑↓ R	1	R↑↑ R	R↑↑ R
2	R↑↓ .	R↑↓ R	1	R↑↑ R	.. R
.	.. R	R↑↓ .	.	.. .	.. .
N-2	R↑↓ R	R↑↓.. R	1	R↑↑ R	R↑↑ . R
N-1	R↑↓	R↑↓ R	1	R↑↑	R↑↑ R

A.2 - 30

Legendă:
↑ = tranziţie forţată din 0 în 1
↓ = tranziţie forţată din 1 în 0
R = citire
0 = scriere 0 (iniţializare)
1 = scriere 1 (iniţializare)

re dată bitul i devine cel mai semnificativ bit în secvenţa de adresare. Partiţionarea memoriei în jumătatea inferioară este înţeleasă în cele de mai jos, funcţie de valoarea bitului de adresă i (Bitul i este rangul c.m. semnificativ al unui numărător). Dacă bitul i din adresa celulei este 0 atunci ea aparţine jumătăţii superioare iar în caz contrar, jumătăţii inferioare. In cele ce urmează ca şi [NAIR78] vom considera biţii de adresă numerotaţi de la 1 la $\log_2 N$.

•Pasul 1. Fie i=1, se iniţializează toate celulele în starea 0.

•Pasul 2. Se execută secvenţele 1 - 8 din tabelul A2.9, partiţionarea memoriei fiind determinată de bitul i de adresă.

•Pasul 3. Se repetă secvenţele 1 - 8 cu deosebirea că în cadrul fiecărei partiţii se face adresarea în ordinea descrescătoare a adreselor.

Pasul 4. Dacă $i=\log_2 N$; i=i+1, salt la pasul 2.

END

25,26. Testele SUK [SUK81]

Conform definiţiei 5 din [SUK81], un element MARCH se notează cu $\overrightarrow{t_1\ t_2\ \ldots\ t_n}$, dacă operaţiile $t_1, \ldots, t_n$ se efectuează asupra fiecărei celule de la C_0 la C_{N-1} şi cu $t_1\ t_2 \ldots t_n$ dacă operaţiile $t_1, t_2, \ldots, t_n$ se efectuează asupra fiecărei celule de la C_{N-1} la C_0.

Operaţii

R - citire a celulei

W - scriere în celulă

W0- scriere de 0

W1- scriere de 1

WC- scriere a complementului stării anterioare (aşteptată sau aparentă).

Cu aceste notaţii, testele SUK se exprimă ca testul A: $\overrightarrow{RW_cW_cW_c}$, $\overleftarrow{RW_cW_c}$, $RW_cW_cW_c$, RW_cW_c; testul B: $\overrightarrow{RW_cRW_cR}_c$, $\overleftarrow{RW_cW_c}$, $RW_cW_cW_c$, RW_cW_c.

[SUK81] demonstrează că testul A, referit în continuare ca MSUKA este un test minimal ce detectează toate defectele de cuplare în absenţa defectelor de tranziţie, celule

TABELUL A2.9 - Secvente utilizate in algoritmul de test NAIR-B

	Adr.	Secventa 1	Secventa 2	Secventa 3	Secventa 4
juma- tate sup.	1	R↓ R	R↓ R	R	R↓ R
	2	R↓ R	R↓ R	(contine R "0")	R↓ R
	..	.. .	.. .	.	.. .
	n/2	R↓ R	R↓ R	R	R↓ R
juma- tate inf.	1	R	R	R↓ R	R
	2	(contine R "0")	(contine R "0")	R↓ R	(contine R "1")
	..	.	.	.. .	.
	n/2	R	R	R↓ R	R

Timp -------->

	Adr.	Secventa 5	Secventa 6	Secventa 7	Secventa 8
juma- tate sup.	1	R	R	R↓ R	R
	2	(contine R "1")	(contine R "1")	R↓ R	(contine R "0")
	..	.	.	.. .	.
	n/2	R	R	R↓ R	R
juma- tate inf.	1	R↓ R	R↓ R	R	R↓ R
	2	R↓ R	R↓ R	(contine R "1")	R↓ R
	..	.. .	.. .	.	.. .
	n/2	R↓ R	R↓ R	R	R↓ R

Timp -------->

blocate şi accese multiple sau toate celelalte defecte (celule blocate, defecte de tranziţie sau accese multiple) dacă nu există defecte de cuplare.

Testul B, referit în continuare ca MSUKB, detectează toate defectele funcţionale dacă nu există accese multiple.

Aplicarea succesivă a testelor A şi B va detecta toate defectele din modelul SUK prezentat în capitolul 2.

Se dă în continuare o interpretare a celor două teste, într-un limbaj ipotetic, cu preluarea din BASIC a buclelor FOR-NEXT.

```
1000 % TESTUL MSUKA
1010 % DURATA = 15*N
1020 FOR I=0 TO N-1 % INF.FUNDAL 0
1030 W0 @ I
1040 NEXT I
1050 FOR I=0 TO N-1 % PRIMUL ELEMENT MARCH
1060 R0 @ I
1070 W1 @ I
1080 W0 @ I
1090 W1 @ I
1100 NEXT I
1110 FOR I=0 TO N-1 % AL DOILEA ELEMENT MARCH
1120 R1 @ I
1130 W0 @ I
1140 W1 @ I
1150 NEXT I
1160 FOR I=N-1 TO 0 STEP -1  % AL 4-LEA ELEMENT MARCH
1170 R1 @ I
1180 W0 @ I
1190 W1 @ I
1200 W0 @ I
1210 NEXT I
1220 FOR I=N-1 TO 0 STEP -1  % AL 5-LEA ELEMENT MARCH
1230 R0 @ I
1240 W1 @ I
1250 W0 @ I
1260 NEXT I
1270 END
1280 % TESTUL MSUKB
1290 %          DEOARECE DUPA EXECUTIA TESTULUI MSUKA MEMORIA
1300 % CONTINE INFORMATIA FUNDAL "0",NU SE MAI INSCRIE INFOR-
1310 % MATIE FUNDAL.
1320 %
1330 FOR I=0 TO N-1  % PRIMUL ELEMENT MARCH
1340 R0 @ I
1350 W1 @ I
1360 R1 @ I
1370 W0 @ I
1380 R0 @ I
1390 W1 @ I
1400 NEXT I
1410 FOR I=0 TO N-1  % AL DOILEA ELEMENT MARCH
1420 R1 @ I
1430 W0 @ I
1440 W1 @ I
1450 NEXT I
1460 FOR I=N-1 TO 0 STEP -1  % AL 3-LEA ELEMENT MARCH
1470 R1 @ I
1480 W0 @ I
1490 W1 @ I
1500 W0 @ I
1510 NEXT I
1520 FOR I=N-1 TO 0 STEP -1  % AL 4-LEA ELEMENT MARCH
1530 R0 @ I
1540 W1 @ I
1550 W0 @ I
1560 NEXT I
1570 END
```

A N E X A A.3

DIAGRAMELE SHMOO (SHMOO PLOTS)

Un circuit integrat de memorie are între 1 şi 3 tensiuni de alimentare; pentru fiecare dintre semnalele de intrare, datele de catalog prevăd anumite limite pentru poziţia fronturilor ridicătoare sau coborîtoare; de asemenea se garantează un timp de acces minim. Vom numi toate aceste variabile parametri. Pentru a putea stabili un program de test eficient, este necesară cunoaşterea interdependenţei dintre aceşti parametri.

Una din căile cele mai frecvent utilizate o constituie diagramele SHMOO. Acestea au fost utilizate iniţial, la memoriile cu ferite, pentru a determina influenţa combinată a curenţilor de inhibiţie şi scriere/citire asupra funcţionării memoriilor.

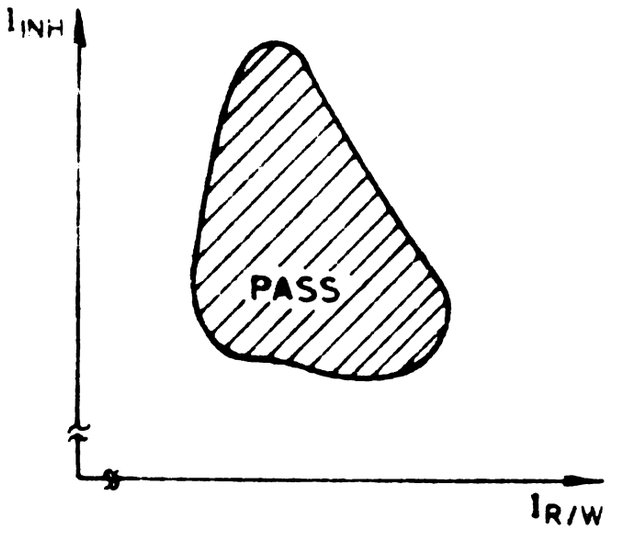


Fig.A3.1. Diagrama SHMOO.

In plan se aleg două axe, pentru curentul de scriere/citire (abcisă) şi pentru curentul de inhibiţie (ordonată).

Conturul care închide toate punctele ($I_{R/W}$, I_{INH}) în care memoria funcţionează (răspunde corect la un test dat), constituie o diagramă SHMOO (fig.A3.1).

Se povesteşte că un nespecialist, privind cîndva o astfel de diagramă, ar fi afirmat că seamănă cu Shmoo, un personaj dintr-o serie de benzi desenate de Al.Capp, care avea succes în epoca respectivă. Astfel acest nume a rămas [FE.78].

Prin extensie, denumirea este utilizată azi pentru toate reprezentările de acest tip, indiferent de parametrii re-

prezentaţi pe cele două axe. De obicei se reprezintă pe cele două axe două dintre tensiunile de alimentare.

Majoritatea echipamentelor de testare automată au posibilitatea de a efectua astfel de reprezentări.

In continuare se dau două diagrame tipice pentru memorii de 1K, după [FEE78] statice, reprezentînd dependenţa timpului de acces de tensiunea de alimentare, pentru două teste standard MARCH şi GALPAT.

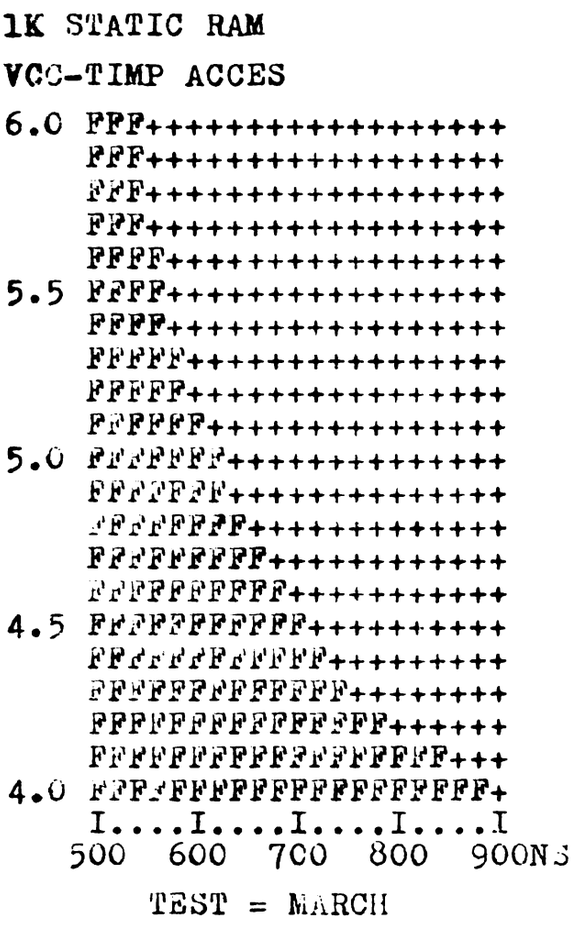


Fig.A3.2.

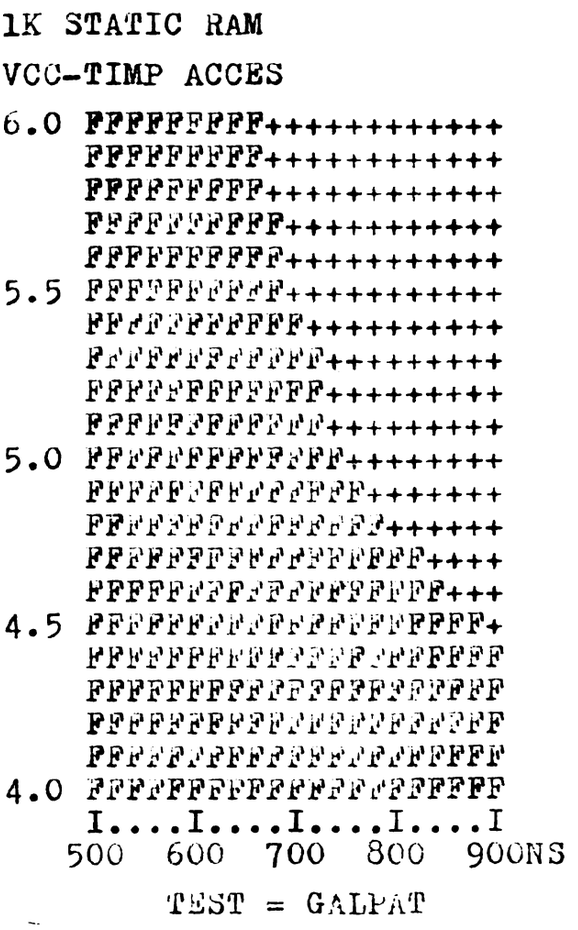


Fig.A3.3.

TABELUL A2.10

TESTE PENTRU MEMORII RAM SEMICONDUCTOARE

Nr. crt.	Test \ defecte	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	Nr.cicluri	Timp de test 4K	16K	64K	UM
1.	CHECKBOARD									+					4N	8.192	32.7	131	ms
2.	MARCH	++			+	+	+			+			+		10N	20.5	81.92	327	ms
3.	MARCH II	++		+	+	+	+			+	+		+		7N	14.33	57.34	229.4	ms
4.	MOVI	++	+	+	+	+	+	+	+	+	+				$N(6\log_2 N+1)$	0.149	0.696	3.18	s
5.	GALCOL	+	+[1]	+	+	++	+								$2(3N^{3/2}+6N)$	0.811	6.39	50.7	s
6.	GALROW	+	+[2]	+	+			++	+						$4(N^{3/2}+2N)$	0.54	4.26	33.8	s
7.	GALDIA	++	++	+											$4(N^{3/2}+2N)$	0.54	4.26	33.8	s
8.	WALPAT	++				+	+								$2(N^2+2N)$	16.8	268	4295	s
9.	GALPAT	++	++	+	+	++	+						++		$2(2N^2+N)$	33.5	537	8590	s
10.	MASPAT		+[3]												10N	20.5	81.92	327	ms
11.	DIAPAT	+	+											+Ψ	$2(2N^{3/2}+2N)$	0.53	4.22	33.7	s
12.	WAKCOL					++								++	$2(N^{3/2}+3N)$	0.274	2.14	16.97	s
13.	DUAL WACOL					++								++	$N^{3/2}+6N$	0.143	1.09	8.58	s
14.	HAMCOL					++								++	$2(N^{3/2}+3N)$	0.274	2.14	16.97	s
15.	HAMPAT					++							++[4]		$2(N^{3/2}+2N)$	0.27	0.12	16.9	s
16.	Dist. WALKING[5]				++										2N(4+8K)	0.114[6]	0.458[6]	1.83[6]	s
17.	X-Y complem.[7]		+												4N	8.192	32.7	131	ms
18.	GWR	++	++	+	+	++	+				++		+		$2(2N^2+N)$	33.5	537	8590	s
19.	Parit.adr.	+								+					4N	8.192	32.7	131	ms
20.	CHEKOL	+				+							+		$2(\frac{1}{4}N^{3/2}+\frac{1}{4}N^2+2N)$	4.26	67.66	1078	s
21.	Knaizuk	+													4N	8.192	32.7	131	ms
22.	Knaizuk-Nair	++													4N	8.192	32.7	131	ms
23.	MILPAT	++			++							++			obs. 8	0.52[9]	1.04[9]	2.08[9]	s
24.	NAIR-A	++			++	++	++	++	++	+	+		++		30N-8	61.4	24.5	983	ms
25.	NAIR-B	++	+	+	++	++	++	++	++	+	+		++		$N(1+32\log_2 N)$	0.78	3.67	16.1	s
26.	MSUKA+MSUAB	++	+	+	+	+	+	+	+	+	+		++		15N+16N=31N	30.72	122.9	460	ms
27.	Refresh											++			funcție de algoritm				

Legendă: ++ testul descoperă cu probabilitate mare defectul respectiv,
+ testul descoperă cu probabilitate mică defectul respectiv,
N = numărul de cuvinte.

1 ++ pentru defecte ale decodificatorului de adresa de coloană.

2 ++ pentru defecte ale decodificatorului de adrese de rînd.

3 ++ pentru defecte statice (permanente) ale decodificatoarelor de adresă.

4 Descoperă scrieri multiple în cadrul aceleiaşi coloane.

5 Testul mai este denumit şi SURROUND DISTURB.

6 Timpii sînt calculaţi pentru K=3.

7 Testul mai este denumit şi ADDRESS COMPLIMENT.

8 $T=N.T_c+2.\frac{N}{N_R-1}.T_R$.

9 Timpii sînt calculaţi pentru T_R=4 ms

Defecte:

F1 = Neîndeplinirea condiţiei de unicitate a adresei.
F2 = Viteză nesatisfăcătoare a decodificatoarelor de adresă.
F3 = Perturbări ale celulei.
F4 = Perturbări între celulele adiacente.
F5 = Perturbări în cadrul coloanei.
F6 = Perturbări între coloanele adiacente.
F7 = Perturbări în cadrul rîndului.
F8 = Perturbări între rînduri adiacente.
F9 = Sensibilitate la informaţie.
F10 = Revenire după scriere.
F11 = Defecte de regenerare.
F12 = Scrieri multiple.
F13 = Revenirea amplificatoarelor de citire.

1900 TIMIŞOARA ● Str. Gh. Lazăr nr. 9 ●
Tel. 961 – 3 00 78 ● Telex : 71380 memor

ANEXA A4

ECHIPAMENT DE TESTARE AUTOMATA A PLACHETELOR DE MEMORIE - MOSTEST-03 D -

SPECIFICATIE TEHNICA

Testorul este destinat depanării, verificării şi testării finale a memoriilor MOS statice şi dinamice de capacitate maximă 1 M cuvînt de 1 - 24 biţi. Prezenţa unei erori se semnalează prin afişarea mesajului şi locaţiei circuitului de memorie defect pe display sau tipărire pe teletype. Sistemul de test este condus de un microcalculator.

Sistemul de test cuprinde:

- Testorul propriu-zis montat într-un dulap metalic ce conţine un sertar cu electronică pe soclurile de memorii, sursă alimentare testor şi sursă programabilă alimentare memorie de testat.

- Cuplor de interfaţare memorie de testat - testor care are o parte comună cu cuplorul de interfaţare memorie de testat.

- Unitate de disc flexibil pentru stocarea programelor de sistem şi de test.

- Dispozitiv de intrare-ieşire date de tip teletype şi display paralel.

- Masă suport.

- Există posibilitatea conectării suplimentare a unui lector şi perforrator de bandă.

Legătura între dulap, cuplor, unitate de disc, teletype, display se realizează prin cabluri flexibile, prevăzute cu cuple.

Descriere funcţională

Din punct de vedere logic sistemul de test cuprinde următoarele părţi

1. Microcalculator pe bază de microprocesoare INTEL 8085, ca unitate de comandă a sistemului de test, avînd ca periferice memorie externă pe unitate de disc flexibil şi dispozitiv de intrare-ieşire de tip teletype şi display paralel. Microcalculatorul este prevăzut cu un sistem de operare software care permite realizarea funcţiilor testorului grupate astfel dialogul operator-sistem, execuţie program de test, tratarea erorilor, crearea şi modificarea programelor de test.

2. Bază de timp cu numărătoare care generează nouă impulsuri programabile cu un număr total de 2o fronturi. Semnalele de timing pot fi programate cu o rezoluţie de 5 ns şi precizie de ±3 ns în domeniul 2oo ns - 5ooo ns.

3. Generator de coduri ce permite verificarea memoriei din punct de vedere al sensibilităţii la coduri de date şi adrese. Generatorul de coduri este realizat în tehnica microprogramată. El este format din următoarele blocuri

- bloc de memorie microprograme
- generator de adrese
- bloc adresare topologică
- generator date
- bloc de comparaţie şi înregistrare a erorii
- memorie rapidă, captoare de erori.

Generatorul de coduri realizează următoarele coduri de testare

Coduri de depanare:

- cod verificare existenţă căi
- cod verificare existenţă căi adrese AFP

Coduri rapide specifice

- tabla de şah - Checkerboard
- paritate
- cod MASEST
- cod MARCH
- cod interacţiune căi date
- Galloping pe linii
- Gallopping pe coloane
- Gallopping pe diagonală

Coduri de regenerare

- cod regenerare statică
- cod regenerare dinamică

Coduri de performanţă

- Gallopping pattern
- Gallopping write-recovery
- Walking

Pe lîngă aceste coduri mai pot fi implementate şi alte coduri specifice memoriei de testat.

Frecvenţa maximă a generatorului de coduri este de 8MHz

4. Surse programabile.

Blocul de surse programabile furnizează - 3 tensiuni astfel

U1= -(0+10V) de 2A programabilă cu pas de 1o mV

U2= +(0+10V) de 10A programabilă cu pas de 1o mV

U3= +(0+20V) de 10A programabilă cu pas de 2o mV

5. Cuplorul ce permite interfaţarea testorului propriu-zis cu memoria de testat. Cuplorul asigură recepţia, emisia, adaptarea şi strobarea semnalelor. El are o parte specifică tipului de memorie testată.

Caracteristici generale

- Gabaritul testorului propriu-zis 6oo x 6oo x 15oo mm^3
- Alimentare la 22o V, 5o Hz
- Greutate cca. 2oo kg
- Consum de energie electrică sub 1ooo VA

- Temperatura de funcţionare $22^{o}C + 3^{o}C$

La solicitările beneficiarului, pe baza specificaţiei tehnice a memoriei se realizează un cuplor specific şi programe de test complete care urmează să se stocheze pe unitatea de disc flexibil.