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

Ing. CHELCEANU MIRCEA - DOREL

COMANDA IN SISTEMELE ELECTRONICE NUMERICE

Conducător ştiinţific:
Prof.dr.ing. MURESAN TIBERIU

TIMISOARA - 1986

PREFATA

Lucrarea de faţă abordează o tematică în concordanţă cu direcţiile şi orientările de perspectivă ale producţiei româneşti privind proiectarea şi realizarea unor sisteme numerice complexe, inclusiv calculatoare şi sisteme de calculatoare într-o concepţie modulară şi multifuncţională în concordanţă şi cu realizările şi perspectivele pe plan mondial.

Această lucrare este o continuare a activităţii autorului în cadrul catedrei de Automatică şi Calculatoare a facultăţii de Electrotehnică din Timişoara, în domeniul proiectării şi realizării unor structuri de comandă complexe, cu multe nivele ierarhice, cu execuţii paralele.

In cadrul lucrării sînt abordate atît probleme teoretice privind structurile de comandă numerice,cît şi aspecte practice legate de concepţia, proiectarea, realizarea şi utilizarea acestora.

In perioada elaborării tezei am beneficiat de îndrumarea competentă a prof.dr.ing.Al.Rogojan, a prof.dr.ing.V.Pop,Conf. dr.ing. C.Strugaru.

In perioada finalizării tezei am beneficiat de sprijinul competent şi îndrumarea atentă şi exigentă a conducătorului, prof.dr.ing.Tiberiu Mureşan căruia îi exprim întreaga mea recunoştinţă.

Tin să mulţumesc colegilor din catedră,pentru sprijinul pe care mi l-au acordat, pentru discuţiile utile, pentru încurajarea morală, dintre care vreau să amintesc pe ş.l.dr.ing. M.Vlăduţiu, ş.l.ing.M.Stratulat, ing.A.Mihăilescu.

Pentru sprijinul moral şi înţelegerea de care a dat dovadă, vreau să mulţumesc în mod deosebit mamei mele.

Calitatea tehnică a materialului redactat (parte desenată, parte dactilografiată) se datoreşte doamnelor M.Dumitrov şi Predoi R., cărora vreau să le mulţumesc şi pe această cale.

Mulţumesc în final, tuturor prietenilor, rudelor care m-au sprijinit, şi au făcut posibilă apariţia acestei lucrări, fără a putea să-i menţionez pe toţi din lipsa spaţiului.

Autorul,

C U P R I N S

Pag.:

ABREVIERI

1 - SN = sistem numeric
2 - CN = calculator numeric
3 - DC = dispozitiv de comandă
4 - SD = secţiune de date a unui sistem numeric
5 - SC = secţiunea de comandă a unui sistem numeric
6 - AsS = automat secvenţial sincron
7 - GSU = generatorul stării următoare
8 - RS = registrul de stare
9 - LFE = logica funcţiilor de ieşire
10 - MUX = multiplexor
11 - DeMUX = demultiplexor
12 - DCD = decodificator
13 - LE = lumea exterioară
14 - M = master (principal)
15 - S = sclave (subordonat)
16 - VID = variabile introduse în diagramă
17 - PLA = Programmable Logic Array (reţea logică programabilă)
18 - DCLM = dispozitiv de comandă locală-memorie

COMANDA IN SISTEMELE ELECTRONICE NUMERICE

1. INTRODUCERE

In literatura de specialitate, abundă titlurile referitoare la dispozitivele aritmetice, dispozitivele de memorie sau cele referitoare la echipamentele periferice. Dispozitivul de comandă (DC) al unui calculator numeric (CN) sau în general al unui sistem numeric (SN) este cel mai puțin reprezentat deși funcționarea sa este mai complexă decît a celorlalte dispozitive. Motivul acestei slabe reprezentări este multitudinea de variante funcționale sau constructive ale DC și lipsa unei metode unitare și general valabile pentru descrierea și proiectarea oricărui tip de DC.

In prima parte a acestei lucrări, se caută a se sistematiza clasificarea, descrierea și proiectarea DC pentru CN sau orice tip de SN.

In continuare se prezintă metode concise și riguroase de proiectare, adaptate de autor după metodele clasice de proiectare.

In sfîrșit, în lucrare sînt descrise metode de segmentare și cuplare a unor DC sau segmenți din unul sau mai multe DC, cu exemplificări practice pe instalații numerice complexe, cu funcțiuni multiple, multe din aceste funcțiuni executîndu-se paralel sau simultan. Aceste instalații au fost realizate de autor independent sau în colectiv.

Tangențial cu subiectele enumerate, sînt tratate probleme de divergență a comenzii, de convergență, concurență, priorități în cuplarea mai multor DC.

Pentru delimitarea domeniului îngust al prezentei lucrări, în continuare se va defini noțiunea de SN [C5, H7] . In fig. 1.1 este prezentată schema bloc a unui SN, cu cele două secțiuni ale sale, SD - secțiunea de date și SC - secțiunea de comandă. Contextul în care este plasat SN este notat cu LE-lumea exterioară. Prin noțiunea de SN se poate considera un

calculator numeric, un bloc al unui calculator numeric (de exemplu blocul de memorie sau un echipament periferic) un procesor specializat pe prelucrarea semnalelor analogice, un sistem de achiziţie/distribuţie date, etc.

Legătura dintre SN şi LE se face prin conexiunile de intrare X_i şi prin conexiunile de ieşire X_e.

Dacă LE este de tipul numeric sincron cu SN atunci X_i şi X_e pot fi reduse la nivelul unor cuple, sau vor conţine şi sincronizatoare, dacă LE şi SN sînt asincrone unul faţă de celălalt.

Dacă LE este de tip analogic, atunci X_i poate conţine elemente de tipul: traductor, multiplexor analogic, corector de scară, bloc eşantionare-memorare analogică, convertor analog-numeric, etc., iar X_e poate conţine următoarele elemente: convertor numeric analogic, demultiplexor analogic, corector de scară, bloc de eşantionare-memorare analogică, filtru, amplificator sau atenuator (divizor), element de execuţie sau indicaţie.

In fig.1.1, cu ID s-au notat intrările de date şi cu ED ieşirile de date, semnalele esenţiale însă din punctul de vedere al lucrării de faţă sînt:

ci - comenzi interne
si - stări interne
cde - comenzi din exterior
cse - comenzi spre exterior
sde - stări din exterior
sse - stări spre exterior.

In blocul SD pot intra orice elemente cu funcţiuni de memorare sau prelucrare numerică a datelor, ca de exemplu dacă SN este CN atunci în SD intră ca elemente componente: blocul de memorie, unitatea aritmetico-logică, echipamentele periferice, iar SC este dispozitivul central de comandă al calculatorului.

In continuare, din cele două componente ale CN nu vom lua în considerare decît SC.

Dacă SN este considerat un bloc de memorie, de exemplu, atunci SD este memoria propriu-zisă, iar SC este dispozitivul de comandă locală a memoriei.

In semnalele "si" pot fi incluse parţial sau total ID,

ED sau alte date interne din SD (bitul de paritate, semnul unui rezultat, etc.).

Funcţionarea unui SN este descrisă într-o primă etapă prin "protocolul de operare" - de fapt o descriere prin cuvinte. In celelalte etape necesare proiectării SN acesta poate fi descris prin: schemă bloc cu borne de intrare-ieşire; diagramă de stări, cronograme, organigrame, tabele, limbaj simbolic (ex.:AHPL), ecuaţii logice, schemă logică (electronică) .

In sfîrşit, în concluzie, rolul SC în cadrul unui SN este de a dirija funcţionarea secvenţială a SD în aşa fel încît întregul SN în ansamblu să se comporte conform protocolului de operare, adică asigurînd intrarea corectă a datelor (ID), prelucrarea, conversia, memorarea lor - după caz şi ieşirea corectă a mărimilor prelucrate sau stocate (ED).Toate aceste funcţiuni sînt asigurate ţinînd cont atît de evoluţia secvenţială a SD cît şi de semnalele externe SN considerat (cde, sde). Dialogul cu LE este asigurat tot de SC (cse,sse).

Din această analiză s-au desprins două funcţiuni majore ale SC:

- comanda internă a SN,
- asigurarea conlucrării (dialogului) cu LE.

In fig.1.2 este ilustrat la nivel primar, într-o primă etapă dialogul dintre două SN, considerînd cazul cel mai simplu cînd un SN este principal (master-M) şi celălalt este subordonat (slave - S). Comunicaţia între cele două SN este asigurată de patru magistrale pentru:

- adrese
- date
- comenzi
- stări

Magistrala de adrese este necesară doar în cazul existenţei a două sau mai multe SN - slave pentru a asigura selecţia momentană a unuia din ele.

Şi în acest caz, întregul dialog este asigurat de către cele două SC din SN considerate, unul fiind subordonat celuilalt.

In fig.1.9 este prezentată o rețea de SN[C5,H7] la care relația MASTER-SLAVĂ este variabilă, adică la un moment dat oricare din SN poate fi MASTER și celelalte SLAVE.

La nivelul la care se analizează relațiile dintre SN într-o rețea, vom renunța în mod intenționat la restul magistralelor, păstrînd o singură magistrală de comunicație, pe care o numim magistrala de comenzi-stări.

În fiecare SN din rețea, SC trebuie să asigure pe lîngă executarea secvențială a protocolului de operare al SN propriu, și conexiunea și dialogul cu alte SN prin intermediul magistralei de comunicație comenzi-stări.

Cazuri mai complexe de dialog, cu mai multe SN-master-momentan, și mai multe dialoguri simultane, vor fi considerate în următoarele capitole.

Considerînd că în acest prim capitol introductiv s-a delimitat cadrul prezentei lucrări, în continuare vor fi tratate succesiv, probleme de detaliu.

2. ROLUL SECȚIUNII DE COMANDA (SC) IN CADRUL UNUI SISTEM NUMERIC (SN)

În continuare, prin abstractizare se va separa SC de SD și se va plasa într-un context oarecare.

În fig.2.1, SC primește intrările "I", generează ieșirile "Y" și asigură funcțiunea de secvențiere "S".

Mai departe, se va desprinde SC de context și se va considera pur și simplu un dispozitiv de comandă (DC) independent, a cărui proiectare depinde de contextul în care urmează să fie plasat.

De fapt, o descriere riguroasă a funcționării unui DC nu conține decît referiri la cele trei elemente de mai sus "I";"Y";"S".

2.1. Funcția de secvențiere a unui dispozitiv de comandă (DC).

DC analizate în lucrarea de față sînt automate secvențiale sincrone (ASS), indiferent dacă sînt de tipul cablat, microprogramat sau programat. Aceasta înseamnă că funcționează corelat cu o bază de timp proprie sau externă, comandată

de unul sau mai multe generatoare de tact aperente sau trans parente (la DC programate). Impulsul de tact, poate fi el însuşi periodic sau neperiodic, aparent sau transparent (la DC realizate cu monostabile) după cum se va vedea mai departe.

Baza de timp însăşi, poate fi distinctă în afara sau în cadrul DC, sau poate fi inseparabilă în cadrul DC (ex.la DC realizate cu numărătoare).

Din motivele arătate mai sus, rezultă că DC analizate aici funcţionează secvenţial, toate elementele continute în protocolul de operare fiind realizate în etape sau paşi. O etapă sau pas din funcţia de secvenţiere, de acum încolo o vom numi stare.

Tipul de secvenţiere sau succesiunea de stări poate fi de tipul:

- liniar
- ciclic
- ramificat
- buclat
- divergent

In fig.2.2 sînt prezentate segmente de secvente pentru cazurile:

- liniar (dacă DC este cu autooprire-sfîrşit mort);
- ciclic - acelaşi desen (dacă DC este cu revenire la starea iniţială);
- ramificat (dacă DC într-o anumită stare au posibilitatea de a merge pe una din două sau mai multe căi posibile în funcţie de o intrare/condiţie);
- buclat (dacă DC are posibilitatea să parcurgă de un anumit număr de ori, un grup de stări, tot în mod condiţionat);

In dreapta fig.2.2, se prezintă schematic înlănţuirea elementelor de memorare a stării curente - bistabile, monostabile, registru, etc. Se remarcă faptul că toate aceste cazuri există un singur trunchi liniar pentru elementele de memorare.

In fig.2.3, este prezentat un segment dintr-un DC divergent. In acest caz, dintr-o anumită stare, în mod condiţionat sau necondiţionat se trece simultan în două sau mai

multe stări, după care DC va parcurge simultan două ramuri distincte. Este cazul în care DC trebuie să comande simultan două execuţii distincte, ca de exemplu o interfaţă comună mai multor echipamente periferice, sau un procesor care lucrează pe principiul aducerii anticipate a instrucţiunilor din memorie. Un alt exemplu ar putea fi cazul unei interfeţe care poate asigura un dialog-stare simultan cu un dialog-date.

In dreapta fig.2.3, se prezintă schematic înlănţuirea elementelor de memorare a stării curente (ex.: bistabili de stare) Se observă că după un trunchi comun urmează două ramuri distincte. Extrapolînd acest caz, se poate ajunge la o structură complexă arborescentă, ceea ce conduce la situaţia ca într-un moment dat DC să se afle simultan într-un număr oarecare de paşi (stări distincte), fără a fi afectat de hazard sau incertitudine în privinţa stărilor următoare, pe diverse ramuri parcurse simultan.

Orice DC are un sistem de iniţializare, adică pleacă la începutul operării sale, dintr-o stare bine precizată (prima stare).

Tranziţiile dintr-o stare în alta pot fi de tipurile:

- tranziţie directă
- tranziţie condiţionată
- buclă de aşteptare
- tranziţie divergentă

Sfîrşitul unei secvenţe poate fi de tipul:

- sfîrşit mort
- revenire la starea iniţială
- convergenţă cu altă secvenţă.

Necesitatea divergenţei comenzii s-a arătat mai sus. Necesitatea convergenţei apare atunci cînd într-un DC ramificat, trebuiesc să se întîlnească două ramuri oarecare şi să se contopească într-o singură ramură. Dacă s-ar şti întotdeauna care din ramuri se încheie mai rapid, atunci aceasta s-a putea prevedea cu un sfîrşit mort, iar trunchiul comun în DC ar urma după încheierea parcurgerii secvenţei din ramura cea mai lentă dintre cele care trebuie să se întîlnească. Insă, în general nu se cunoaşte ramura cea mai rapidă sau cea mai lentă şi este necesară prezenţa unei scheme de convergenţă.

In capitolele următoare va fi tratată mai în detaliu proble-

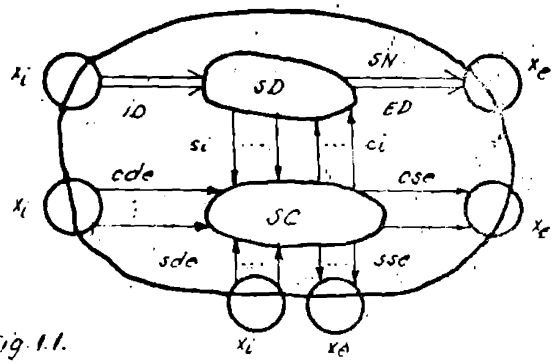


Fig. 1.1.

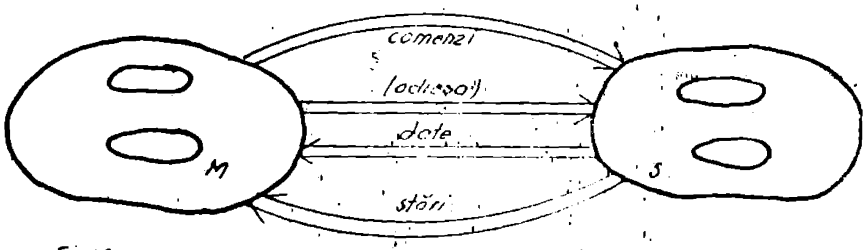


Fig 1.2.

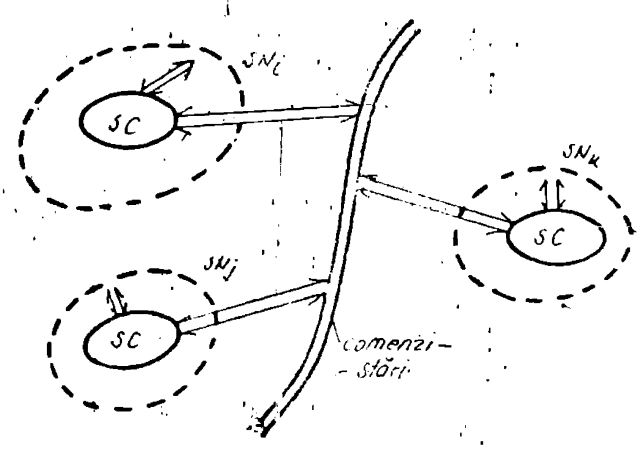


Fig. 1.3.

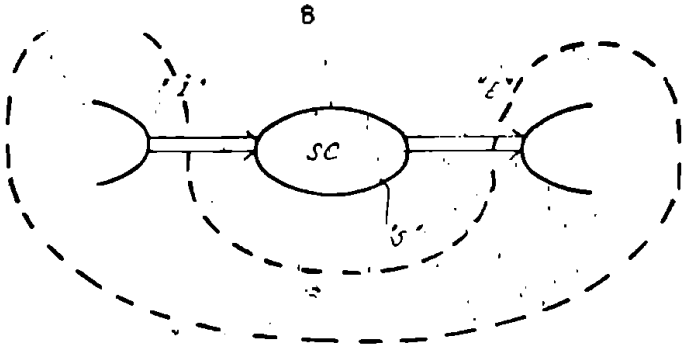


Fig. 2.1.

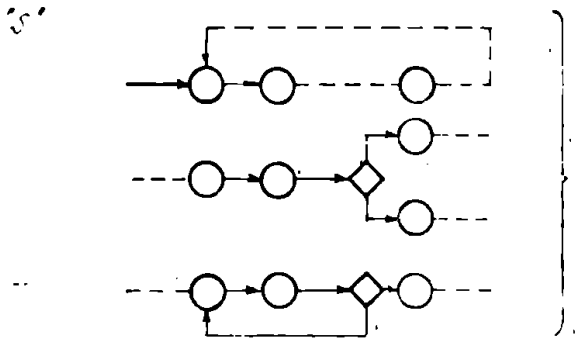


Fig 2.2.

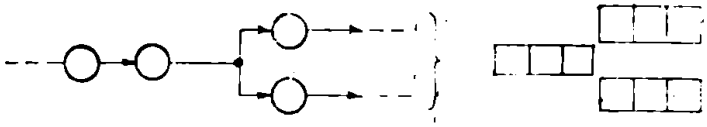


Fig 2.3.

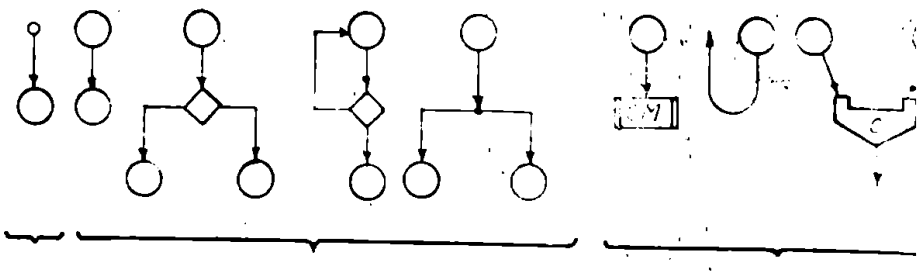


Fig 2.4

ma divergenței/convergenței și se va prezenta o metodă de proiectare a unei scheme universale de convergență.

Foarte schematic, în fig.2.4 sînt prezentate începutul unei secvențe, tranzițiile și sfîrșitul uneia sau mai multor secvențe paralele. S-au utilizat următoarele simboluri:

- cerc = stare
- romb = intrare/condiție
- săgeată = sens tranziție
- punct înterogat = punct de divergență
- dreptunghi cu înălțimile dublate (două linii paralele) = sfîrșit mort (SM)
- figura notată cu "C" = schemă de convergență.

2.2. Condiții sau intrări într-un DC

Într-un DC propriu-zis nu există intrări de date sau de adrese ci numai intrări de condiționare (pe lîngă intrarea de inițializare sau eventual cea de tact).

Dacă anumite date obținute în urma unor prelucrări în SD a unui SN determină secvențele parcurse în SC (DC), la nivelul acestuia, tot intrări de condiționare sînt considerate.

Dacă din mai multe DC, este selectat unul singur care va fi declanșat, cu ajutorul unui cuvînt de adresă, atunci un decodificator distinct de DC va determina care DC este activ. În acest caz fiecare ieșire a decodificatorului va constitui o intrare de condiție pentru cîte un DC. În general, decodificatorul este segmentat și distribuit fizic lîngă DC-urile respective (cazul mai multor interfețe selectate printr-un cod de selecție/adresă).

Intrările într-un DC pot fi de tip impuls sau nivel. Semnalele corespunzătoare intrărilor de tip impuls trebuiesc memorate temporar în bistabili pentru a nu se pierde, dacă perioada de repetiție a tactului DC este mai mare decît durata impulsului.

Intrările pot fi independente, adică nu sînt condiționate în nici un fel de secvențe parcurse de DC, și sînt în general semnale exterioare SN considerat, servind la declanșarea DC, oprirea lui sau pur și simplu, la o ramificare

sau o buclare. In principiu, aceste intrări sînt asincrone față de DC.

Altă categorie de intrări sînt cele dependente de funcționarea DC. De exemplu, un numărător este incrementat/decrementat la comanda DC. Ieșirile numărătorului pot constitui intrări în DC fiind de fapt intrări condiționate. În principiu, acestea sînt sincrone cu DC, excepție făcînd cazurile cînd blocul comandat (ex.numărătorul amintit mai sus) este cu funcționare asincronă (monostabil, unitate aritmetică-logică, linie de întîrziere, etc.).

Indiferent de faptul dacă intrările sînt dependente sau independente, dacă sînt asincrone, ele pot fi preluate de către DC în mod sincron sau asincron. În cazul preluării asincrone, pot apare probleme de hazard static sau dinamic; proiectarea DC este mai dificilă, testarea greoaie.

În cazul preluării sincrone, se utilizează înainte de intrarea în DC sincronizatoare de nivel sau de impuls, după caz.

Concluzionînd, după considerațiile din paragrafele 2.1, și 2.2, funcție de secvențiere "S" este condiționată într-un număr K de stări din totalul de n stări posibile ale DC, de către intrările "I";

unde : $0 \leq K \leq n$

2.3. Ieșiri sau comenzi ale unui DC

Pe lîngă funcția de secvențiere "S" efectată după cum s-a văzut, de către intrările "I", un DC trebuie să asigure generarea ieșirilor "h", care de fapt sînt comenzi pentru secțiunea de date aflată sub controlul DC, sau stări/comenzi pentru alte sisteme numerice cuplate cu SN ce conține DC considerat.

Aceste ieșiri servesc la acționarea SD a SN respectiv, sau sînt semnale ce merg spre alte SN, în cazul mai multor SN cuplate.

Ieșirile pot fi:

- necondiționate: într-o anumită stare se generează o ieșire, cu durata egală de obicei cu perioada de repetiție a tactului, independent de condiționările exterioare "I".

- condiționate: într-o anumită stare se generează o ieșire, în funcție de una sau mai multe intrări de condiție. Evident, în orice stare se poate genera sau nu o ieșire (sau mai multe), condiționat sau necondiționat.

Ieșirile pot fi generate direct (durata egală cu perioada tactului), prin produs logic între o stare și tact (durata egală cu durata tactului), prin intermediul unui monostabil (durata egală cu durata impulsului generat de monostabil pe perioada funcționării sale în regim cvasistabil), sau prin intermediul unui bistabil.

Dacă o ieșire este generată prin intermediul unui bistabil, atunci într-o anumită stare bistabilul este poziționat pe 1, iar în altă stare este șters. Durata acestui tip de ieșiri este de n perioade de tact.

Dacă o ieșire trebuie să fie sincronă cu dispozitivul acționat de ea, atunci aceasta se va sincroniza cu tactul acestui dispozitiv.

Uneori o ieșire se sincronizează cu tactul DC ce o generează (sincronizare la ieșire), din diverse considerente, ca de exemplu pentru a obține o întîrziere de o perioadă de tact în efectul acestei ieșiri.

In cazul generării unei ieșiri prin monostabil, în majoritatea cazurilor este necesară sincronizarea ieșirii monostabilului, pentru ca și frontul său de cădere să fie sincron cu DC.

In cazul ieșirilor condiționate, intrarea de condiționare trebuie sincronizată pentru a evita apariția unor ieșiri cu durate incerte (hazard dinamic).

Cîteva precauții trebuiesc luate, la proiectarea unui DC, și anume:

- buclele de așteptare ale fronturilor de comutare ale unor intrări trebuie să fie plasate în așa fel în secvența DC, încît în mod sigur, în timp, să preceadă aceste fronturi.

- în cazul unui DC divergent, ramurile cu execuții paralele din cadrul DC să aibă resurse hardware distincte din cadrul SC, sau să se prevadă scheme de priorități (concurență).

-starea corespunzătoare unei scheme de convergență să apară în secvența DC într-o astfel de poziție, încît să fie

activă înaintea încheierii execuţiei pe ramura cea mai rapidă din cadrul ramurilor paralele, altfel convergenţa nu va fi niciodată asigurată, şi astfel DC va intra într-un sfîrşit sort fals, neprevăzut în protocolul său de operare, corespunzător buclelor de aşteptare din cadrul schemei de convergenţă (aceste fenomene se vor analiza în detaliu mai tîrziu).

- nu se va testa niciodată o intrare condiţionată, înaintea duratei sigure de stabilizare a acesteia. De exemplu, nu se va testa starea unui numărător, în acelaşi pas din secvenţă în care se incrementează/decrementează numărătorul.

Simbolul unei ieşiri în cadrul unei secvenţe este un dreptunghi.

3. CLASIFICAREA DISPOZITIVELOR DE COMANDA

Întocmai ca orice funcţiune realizată pe cale numerică, şi funcţiunile unui DC pot fi realizate prin mijloace, hardware, firware, software sau combinat. Ţinînd cont de acest lucru, un DC poate fi de tipul:

- cablat
- microprogramat
- programat.

In cazul unui SN complex, sau al unei reţele de SN, aceste trei tipuri de DC, conlucrează. In continuare, vor fi analizate tipurile enumerate mai sus, cu accent pe DC cablate.

3.1. Dispozitive de comandă cablate.

DC cablate sînt dispozitive cu funcţionare automată de tip sincron sau asincron. Cele de tip sincron necesită un impuls de iniţializare, şi impulsuri de tact periodice. Cele de tip asincron, necesită doar un impuls de declanşare (schemele cu linii de întîrziere, circuite basculante monostabile).

Automatele secvenţiale sincrone sînt dispozitive realizate din elemente ca : porţi logice, bistabile, registre, numărătoare, decodificatoare, multiplexoare, memorii fixe (ROM), reţele logice programabile (PLA) .

3.1.1. DC de tip asincron

3.1.1.1. DC realizate cu circuite monostabile

In cazurile în care trebuiesc generate secvențe de impulsuri neperiodice cu durate și întîrzieri oarecare în raport cu momentul inițial și independente de vreun tact se utilizează scheme cu monostabile (ex.:dispozitiv de comandă locală pentru un bloc de memorie).

In fig.3.1 este prezentată organigrama de funcționare și schema unui astfel de DC. Pentru fiecare ieșire a DC (I_{OA},I_{OB}, I_1) se utilizează cîte un lanț din două monostabile, astfel:

- un monostabil asigură întîrzierea față de impulsul de start (M_{OAi}, M_{OBi}, M_{1i}).
- un monostabil asigura formarea în durată a ieșirii respective (M_{OAf}, M_{OBf}, M_{1f}).

Dacă declanșarea unui astfel de lanț este condiționată atunci impulsul de declanșare a lanțului este format printr-un produs logic între intrarea de condiție și impulsul de start. Astfel, la intrarea primului monostabil din lanț apare un circuit "ȘI" (In cazurile mai complexe, o logică combinațională). Dacă pe o singură linie de ieșire, pot apare succesiv în timp, mai multe impulsuri, cu diferite întîrzieri și durate (ex.:I_0 de mai sus) atunci ieșirile corespunzătoare lanțurilor pentru fiecare impuls, se vor reuni într-un circuit "SAU". Dacă și la acest nivel apar condiționări, va apare o logică combinațională de tipul ȘI-SAU.

Cazul analizat mai sus este un exemplu tipic de proiectare a unui DC pe baza unei cronograme. La fel este și cazul următor.

3.1.1.2. DC realizate cu linii de întîrziere

O linie de întîrziere este declanșată cu un semnal de start. Impulsul de start întîrziat cu o cuantă de timp apare la prima ieșire a liniei, cu două cuante de timp la a doua ieșire, etc. în funcție de tipul liniei, cuanta de timp este de la cîteva ns la cîteva sute de ns. Ieșirile liniei se nu-

mesc prize ($P_1 \ldots P_n$). In fig.3.2,a, este prezentată schema bloc a unei linii de întîrziere şi cronograma sa de funcţionare.

O aplicaţie tipică a unui DC cu linii de întîrziere este tot la comanda locală a unei memorii.

Condiţionările ieşirilor în funcţie de intrările de condiţie şi în funcţie de timp sînt similare ca la DC ca monostabile şi porţi logice.

Metoda de proiectare este similară cu cea descrisă în 3.1.1.1. şi pleacă de la cronograma de funcţionare.

Pentru realizarea unui astfel de DC, pe lîngă linia de întîrziere (uneori linii înseriate, conectate în paralel,etc) se mai utilizează porţi logice şi bistabile.

Pentru fiecare bornă de ieşire a DC se utilizează cîte un bistabil care este adus în starea 1 sau 0 în mod condiţionat sau nu, de către impulsurile prizelor liniei.

In fig.3.2,b, este prezentată cronograma unui DC cu trei ieşiri (I_0, I_1, I_2) şi o intrare de condiţie (X), precum şi ecuaţiile de intrare ale unuia din bistabili (I_2), iar în fig.3.2,c, schema corespunzătoare. De remarcat prezenţa intrării de iniţializare $\overline{R}$ (reset).

DC realizate cu linii de întîrziere sînt dintre cele mai rapide însă pentru un protocol de operare complex rezultă o mare risipă de circuite logice.

3.1.2. DC de tip sincron

3.1.2.1. DC clasic pentru un CN [R5,C5]

Majoritatea calculatoarelor din prima şi a doua generaţie au în structura lor un dispozitiv central de comandă de tip sincron care necesită iniţializare, declanşare şi impulsuri de tact cu o anumită frecvenţă de repetiţie.

Scopul unui astfel de DC este să genereze o secvenţă de comenzi pentru aducerea instrucţiunii curente din memorie în registrul de instrucţiuni într-un ciclu de maşină (fază), şi apoi în unul sau mai multe cicluri de maşină să determine execuţia instrucţiunii.

Schema bloc principială a unui astfel de DC este prezentată în fig.3.3.

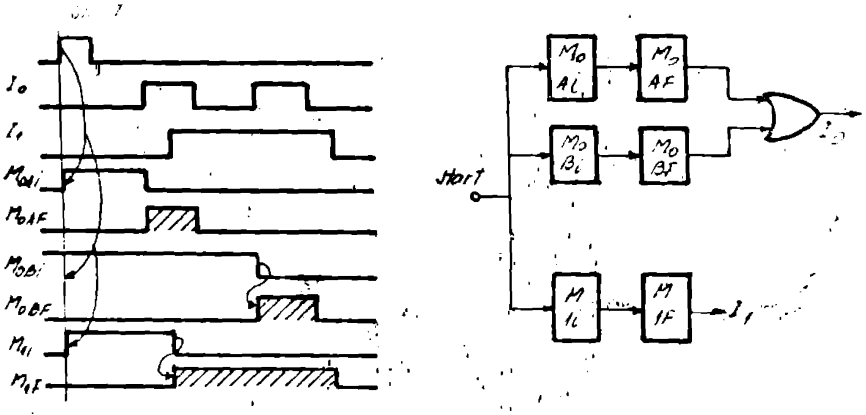


Fig. 3.1

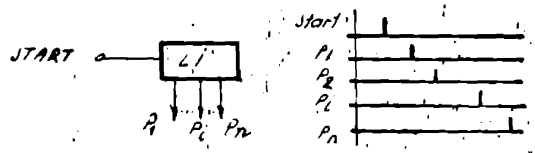
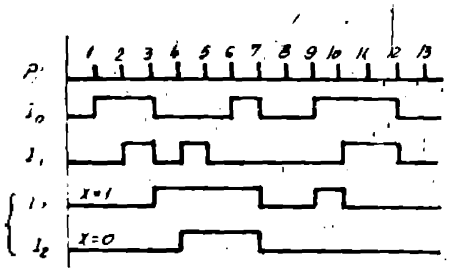


Fig. 3.2.a.



$$J_2 = x \cdot P_4 + x \cdot P_9 + \bar{x} \cdot P_5$$

$$K_2 = x \cdot P_7 + x \cdot P_{10} + \bar{x} \cdot P_8$$

Fig. 3.2.b.

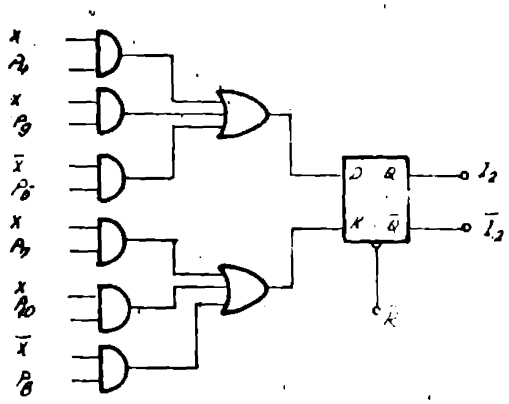


Fig 3.2.c

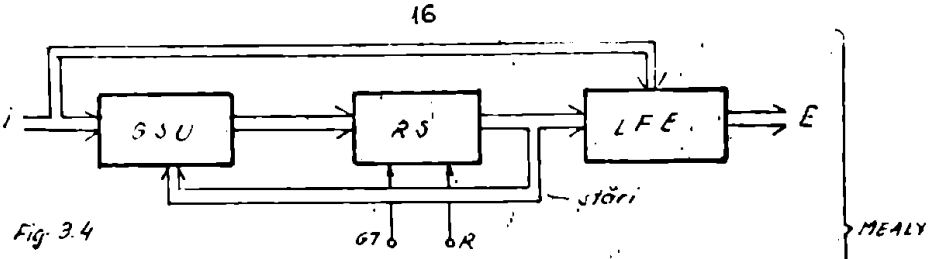


Fig. 3.4

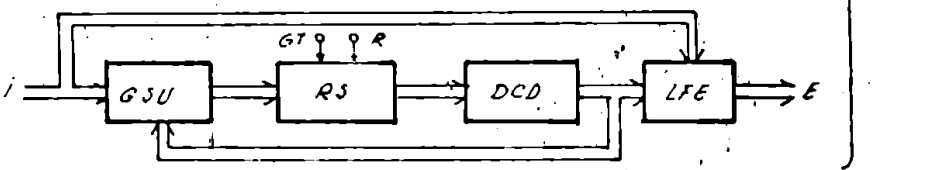


Fig. 3.5

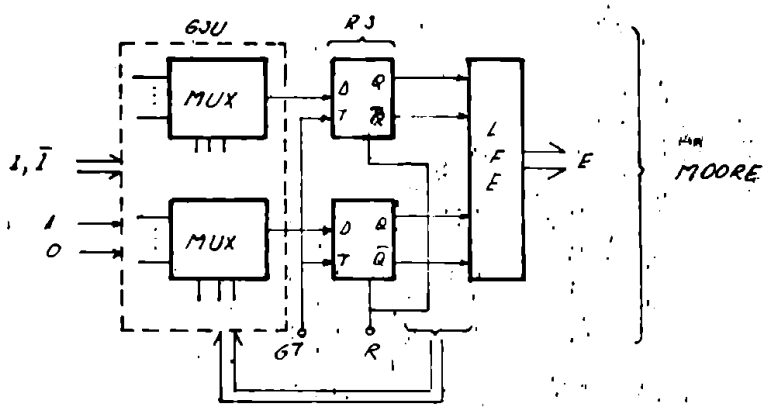


Fig. 3.6

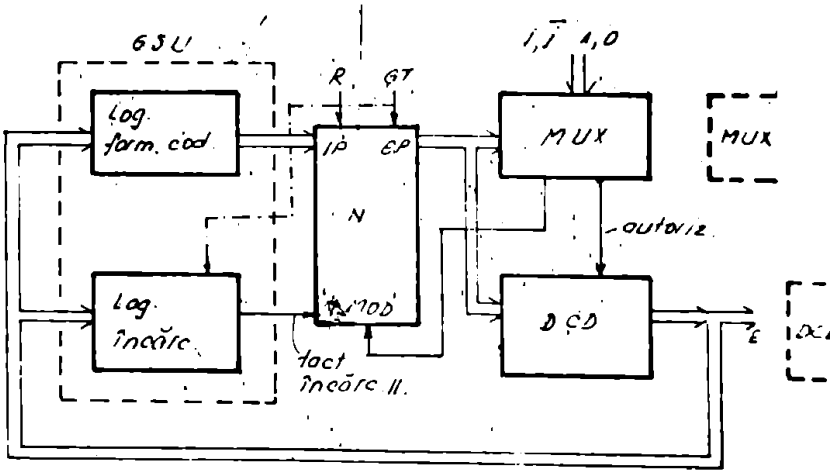


Fig. 3.7

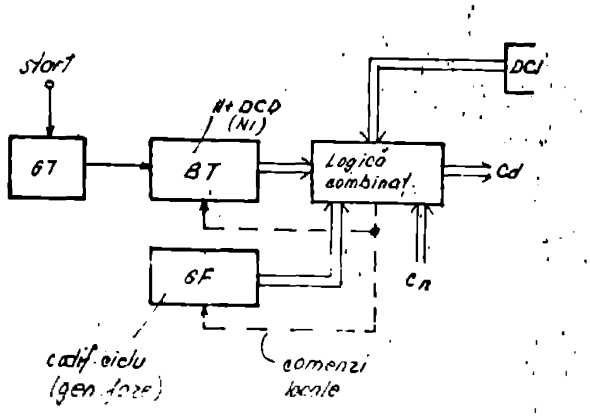


fig. 3.3

Elementele constructive sînt următoarele:

GT - generator al impulsurilor de tact.

BT - baza de timp (generator pentru impulsuri de orologiu) Asigură o secvențiere liniară, ciclică sau bucălată. In realizările mai recente apar și secvențieri ramificate, mai rar cele divergente. Este realizată cu un numărător și un decodificator, sau un registru de recirculare (numărător în inel).

GF - generator de faze. Asigură o secvențiere corectă a ciclurilor de mașină (faze) corespunzător aducerii și execuției fiecărei instrucțiuni. GF comută într-o nouă stare la sfîrșitul fiecărui ciclu al BT. Avem deci două secvențiatoare distincte, cu funcționare simultană, cuplate între ele, dar neconcurente.

Logica combinațională - pe lîngă asigurarea tranzițiilor corecte în cele două secvențiatoare, mai asigură generarea comenzilor Cd (ieșiri din DC) spre SD a CN ținînd cont de următoarele:

- starea curentă a BT
- starea curentă a GF
- codul instrucțiunii curente (ieșirea DCI- decodificatorul instrucțiunii).
- condiționările Cn (semnale de stare din restul CN, adică SD a CN; biți de date; semnale de derutare; cereri de întrerupere interne sau externe; cereri de acces direct la memorie; comenzi manuale, etc.).

517.399
359E

Cele mai utilizate metode de proiectare pentru întreaga logică combinațională sînt metodele tabelare.

Pe baza tabelelor se scriu ecuațiile logice ale funcțiilor de ieșire (comenzi) și ale funcțiilor ce asigură secvențierea corectă în RT și GF.

In expresia logică a unei astfel de funcții, într-un termen oarecare există un produs logic între: starea RT, starea GF, ieșirea DCI și un număr de 0 - n condiționări.

Acest tip de DC poate fi folosit și în alte scopuri decît pentru comanda centrală a unui CN. In acest caz va lipsi DCI, și poate lipsi GF.

Intrucît manipularea tabelelor amintite mai sus este greoaie și laborioasă, în prezenta lucrare se va încerca elaborarea unei metode matriciale pentru proiectarea logicii combinaționale, metodă ce se pretează la proiectarea automată, cu ajutorul unui CN.

3.1.2.2. Automate secvențiale sincrone cu stări codificate [C10, M5, E8, S6, S8]

ASS este cel mai utilizat DC, datorită mai multor considerente:

- poate opera practic la orice frecvență de repetiție a impulsului de tact, o limitare superioară existînd în domeniul zecilor sau sutelor de MHz, funcție de tipul circuitelor utilizate;
- proiectarea sa este simplă și sistematică
- asigură o testabilitate ridicată.
- dacă regulile de proiectare se respectă strict, nu apar fenomene de hazard dinamic nici dacă intrările sînt de tip impuls, sau asincrone.

Schema bloc a unui astfel de DC este prezentată în fig. 3.4. Elementul esențial este RS - registrul de stare. Acesta este inițializat cu comanda R și execută tranzițiile de stări în mod sincron cu tactul CT. RS este format din n bistabile (D sau JK), DC putînd avea maximum 2^n stări distincte, codificate binar pe aceste bistabile. Funcția de secvențiere este asigurată de GSU - generatorul stării următoare (un grup de scheme combinaționale, eventual cu sincronizatoare pentru intrări). Fiecare tranziție este dependentă de starea curentă

şi dependentă sau nu de una sau mai multe intrări.

Funcţiile de ieşire sînt generate de LFE - logica funcţiilor de ieşire (scheme combinaţionale, urmate uneori de monostabile - pentru diverse temporizări nemultiplii a perioadei de tact, sau bistabile pentru sincronizare la ieşire sau asigurarea unei durate a funcţiilor de ieşire - multiplii a perioadei de tact).

Proiectarea acestui tip de DC este aproape standardizată şi conţine următoarele etape:

- pe baza protocolului de operare se întocmeşte organigrama de stări cu următoarele simboluri: cerc = stare, romb= intrare; dreptunghi= ieşire, săgeată = tranziţie.

- se întocmeşte diagrama Veitch (Karnaugh) a stărilor curente urmărind adiacenţe între stări succesive. Codurile binare ale stărilor, rezultate din diagramă, se trec şi pe organigramă.

- pentru fiecare intrare de bistabil din RS respectiv pentru fiecare funcţie de ieşire se întocmeşte cîte o diagramă utilizînd tehnica VID (variabile introduse în diagramă).

- pentru fiecare diagramă se scrie ecuaţia corespunzătoare, sub formă de sumă de produse logice.

- pe baza ecuaţiilor se vor sintetiza GSU şi LFE.

O variantă a DC de mai sus, este prezentată în fig.3.5. Singura modificare este apariţia decodificatorului DCD, ce decodifică cuvîntul de stare. Acest fapt prezintă două aspecte:

- simplifică LFE şi GSU.

- micşorează limita superioară a frecvenţei de repetiţie a impulsurilor de tact.

Pentru a asigura un compromis optim între aceste două aspecte contradictorii se poate utiliza DCD parţial pentru un cîmp al cuvîntului de stare restul rămînînd codificat.

O altă variantă, realizează GSU cu multiplexoare - MUX (fig.3.6). Pentru fiecare intrare de bistabil D din RS se utilizează cîte un MUX de tipul 4/1, 8/1, etc. La intrările de selecţie ale MUX este adus cuvîntul din registrul de stare, iar la intrările de date ale MUX sînt realizate conexiuni la 1 logic, 0 logic, o intrare oarecare I, intrarea inversată $\overline{I}$.

Proiectarea se face pe baza unui tabel de descriere a tranzitiilor de stări (necondiționate - 1,0) sau condiționate (I , $\overline{I}$).

Intr-o anumită stare (intrarea de selecție), un bistabil oarecare din RS va trece în 1 (intrare 1) sau în 0 (intrare 0) în mod necondiționat, sau într-o valoare logică identică cu o anumită intrare (intrare I), sau cu negata ei (intrare $\overline{I}$) în mod condiționat.

Tabelul se întocmește pe baza organigramei de stări a DC.

Metoda prezintă avantajul că în cazul unei implementări cu circuite integrate, se va reduce substanțial numărul de circuite integrate din OSU.

In fig.3.7 este prezentată o variantă de ASS cu stări codificate ce utilizează circuite integrate specializate (numărător, decodificator, multiplexor).

La acest tip de ASS, codurile succesive ale stărilor din porțiunea liniară a secvenței trebuiesc să fie identice cu codurile ce rezultă succesiv la ieșirea numărătorului prin incrementare. In cazul ramificațiilor, codul stării următoare este încărcat paralel în numărător. In acest caz prin intrarea MOD numărătorul este comandat să treacă din regimul de numărare în regimul de încărcare paralel.

OSU este format din două grupe de circuite combinaționale, unul pentru formarea codului stării următoare în cazul ramificațiilor și unul pentru formarea funcției logice a tactului de încărcare paralel.

Modul de lucru al numărătorului este funcție de starea curentă și condiționările exterioare, prin intermediul MUX care primește la intrările de selecție, ieșirea paralel a numărătorului (EP), iar la intrările de date, ca în cazul anterior 1, 0, I, $\overline{I}$.

Pentru generarea funcțiilor de ieșire, se utilizează un decodificator conectat la ieșirea numărătorului.

Validarea sau autorizarea unei anumite funcții de ieșire într-o anumită stare este realizată tot de MUX în funcție de starea curentă și condiționări.

In cazul ramificațiilor înlănțuite (ramuri din ramuri) se

pot utiliza în continuare alte perechi MUX-DCD.

Proiectarea acestui tip de DC pleacă fie de la tabele fie de la organigrama stărilor.

3.1.2.3. Automate secvențiale sincrone realizate cu memorii ROM [M5, P8, S6]

Pot fi cu stări codificate total sau, parțial codificate, parțial necodificate.

Cea mai simplă schemă posibilă conține o memorie ROM, un registru de stare RS și blocul LFE (fig.3.8).

RS și LFE au funcțiunile deja analizate în 3.1.2.2.

Memoria ROM (PROM) servește la memorarea codurilor tuturor stărilor din organigrama de stări.

La intrarea de adrese a memoriei ROM se prezintă un cod de adresă format din două cîmpuri:

- cîmpul format din cuvîntul de stare din RS;
- cîmpul format din codul valorilor momentane al tuturor intrărilor.

La ieșirea de date a memoriei ROM va apare codul stării următoare, deci în funcție de starea curentă și de intrări.

Acest tip de DC se utilizează în cazul protocolurilor de operare complexe cu foarte multe stări, și atunci cînd viteza de execuție nu este un factor esențial.

Alte variante, utilizînd în plus, multiplexoare, numărătoare, bistabile, decodificatoare, sînt prezentate în fig.3.9 3.14.

La proiectarea acestor tipuri de DC, se pot utiliza aceleași metode (organigrame de stări, tabele).

Schema din fig.3.9 conține în plus față de cea din fig. 3.8 două multiplexoare MUX1 și MUX2.

MUX1 selectează intrarea care într-o anumită stare determină tranzițiile spre stările următoare.

MUX2 selectează una din două noi stări posibile fiind comandat de MUX1.

In acest caz lungimea de cuvînt a memoriei ROM este dublă față de cazul anterior, dar este de capacitate mai mică (număr mai mic de cuvinte).

In schema din fig.3.10 se utilizează principiile de la schema din fig.3.6 cu diferența că pe traseul RS-multiplexoa-

re, se intercalează o memorie ROM pe post de convertor de cod. Memoria ROM primeşte la intrarea de adresă codul din RS iar la ieşirea de date furnizează pe mai multe cîmpuri, codurile de selecţie ale multiplexoarelor care au exact acelaşi rol ca multiplexoarele din fig.3.6 (GSU).

In fig.3.11, se utilizează o soluţie de compromis între schemele din fig.3.8 şi 3.10. Un cîmp de n-1 biţi din cei n ai cuvîntului de stare este citit din ROM. Al n-lea bit al noului cuvînt de stare este cel determinat de intrări. Deci, dacă cele două noi stări posibile diferă doar prin acest bit înseamnă că în organigrama de stări ele trebuiesc să fie codificate adiacent. Fiecare cuvînt din ROM are două cîmpuri un cîmp pentru generarea celor n-1 biţi ai noului cuvînt de stare şi un cîmp de m biţi pentru intrările de selecţie ale multiplexorului ce determină selectarea intrării care într-o anumită stare determină tranziţia spre două stări următoare posibile, adiacente. Numărul de intrări trebuie să fie:

$$I \leq 2^{m}$$

In fig.3.12, apare o modificare faţă de fig.3.11 în sensul că RS este realizat cu ajutorul unui numărător cu posibilitatea de numărare în sensul determinat de succesiunea de stări de pe porţiunea liniară a organigramei de stări. Ramificaţiile sînt determinate ca şi buclările prin încărcarea în paralel a numărătorului cu codul noii stări. Acest cuvînt paralel este citit din ROM, la fel bitul ce determină modul de funcţionare al numărătorului.

Tactul de încărcare şi tactul de numărare sînt generaţi astfel:

- unul este generat ca un bit citit de ROM ;
- celălalt este generat ca un bit funcţie de intrări, selectarea MUX fiind făcută cu un cîmp al cuvîntului citit din ROM.

Schema din fig.3.13 este asemănătoare schemei din fig. 3.9 cu diferenţa că selectarea MUX afectat intrărilor este asigurată nu de RS ci de un cîmp al cuvîntului citit din ROM, şi pentru eliminarea fenomenelor de hazard datorită intrărilor asincrone, între cele două multiplexoare, se utilizează un sincronizator de nivel realizat cu bistabil de tipul D.

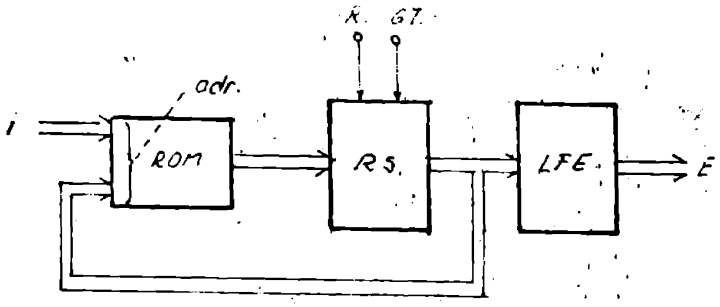


Fig. 3.8.

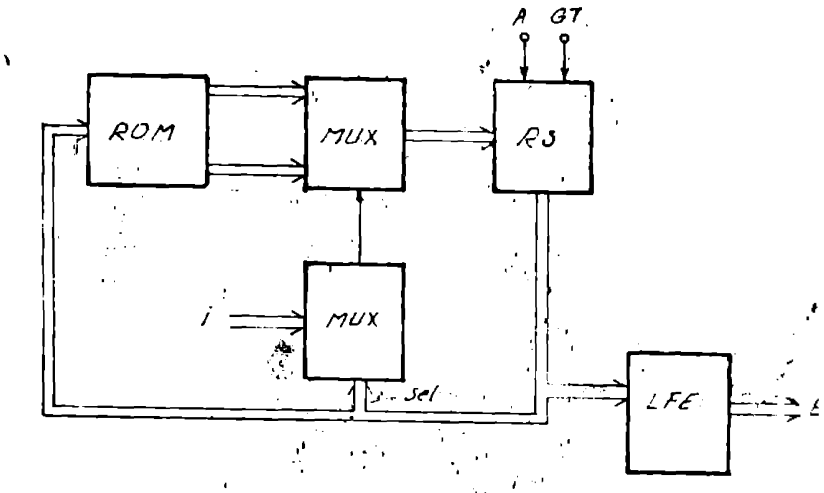


Fig. 3.9

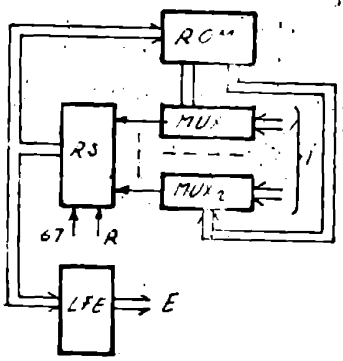


Fig. 3.10

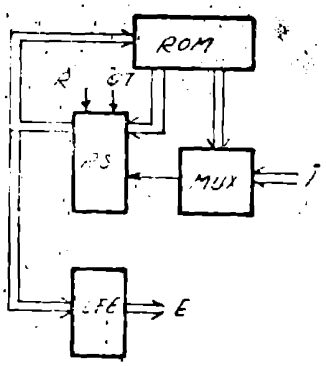


Fig. 3.11

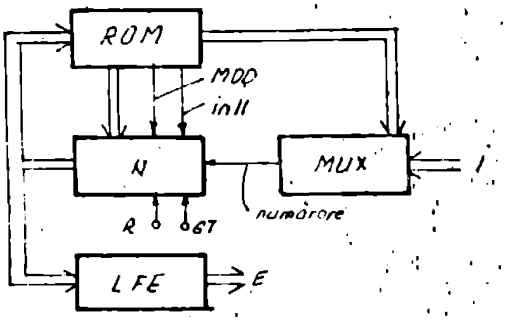


Fig. 3.12.

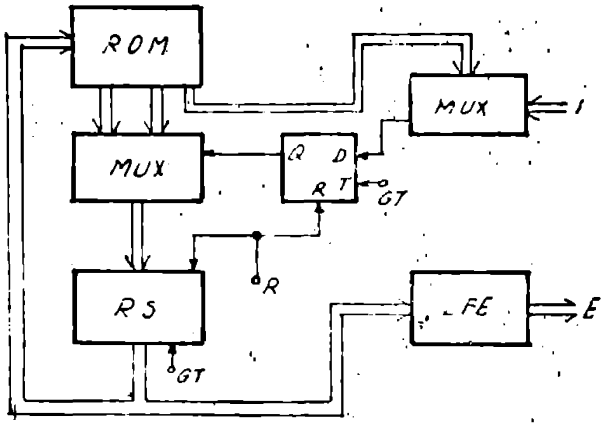


Fig. 3.13.

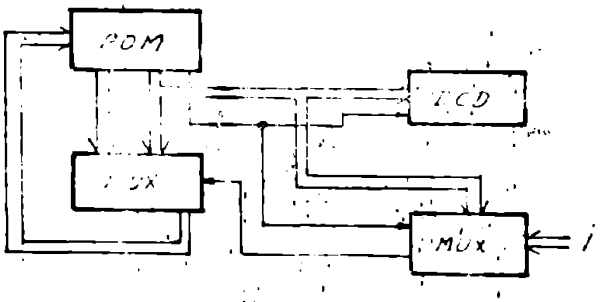


Fig. 3.14.

In sfîrşit, în fig.3.14 este prezentată o schemă fără registru de stare, rolul acestuia fiind preluat de memoria ROM. Cuvîntul de la ieşirea ROM este fix pînă la comutarea intrării selectate de MUX afectat intrărilor. Generarea ieşirilor se face prin decodificarea unui cîmp al cuvîntului citit din ROM, validarea lor făcîndu-se cu un bit citit tot din ROM. Validarea MUX afectat intrărilor se face tot cu ajutorul acestui bit.

In concluzie, alegerea uneia din variantele de DC ce conţin o memorie ROM, se face în funcţie de numărul de stări intrări şi ieşiri, lungimea cuvîntului memoriei ROM disponibile, capacitatea memoriei ROM disponibile, viteza de operare, numărul de circuite integrate utilizate, preţul de cost.

3.1.2.4. Dispozitive de comandă realizate cu reţele logice programabile (PLA) [A4, C3]

In fig.3.15 este prezentată o schemă bloc a unui DC realizat cu un modul PLA. La o analiză atentă, se observă că această schemă este similară schemei din fig.3.4. Diferenţa constă în faptul că atît GSU cît şi LFE sînt implementate cu un singur modul PLA. Intrările în PLA sînt constituite din două categorii de informaţii:

- condiţionările exterioare
- starea automatului (ieşirile RS).

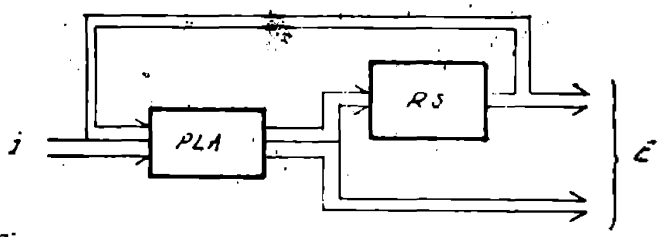


Fig. 3.15

Deci PLA operează asupra unor intrări independente cît şi asupra unora dependente.

Funcţia de secvenţiere a automatului este asigurată de o categorie de ieşiri ale PLA, cele care sînt conectate la intrările bistabililor din registrul de stare.

Altă categorie de ieşiri ale PLA, constituie ieşiri din

automat (comenzi). Celelalte ieşiri din automat, sînt tocmai ieşirile MS.

Acest tip de automat poate fi considerat de tip MOORE sau MEALY după cum se consideră o categorie de ieşiri sau cealaltă categorie. Intradevăr o categorie de ieşiri depinde numai de starea curentă a automatului, iar cealaltă categorie depinde atît de starea curentă cît şi de intrări (ieşirile generate direct de PLA).

Pentru a putea analiza mai în detaliu, probleme realizării unui DC cu PLA, se prezintă în fig.3.16 o schemă tipică de PLA. Blocurile sale componente sînt:

- inversoarele de intrare - pentru fiecare intrare cîte un inversor.

- matricea produselor logice - un număr de n circuite SI fiecare cu atîtea intrări cît este dublul numărului de intrări în PLA. Conectarea intrărilor directe sau negate ale PLA la intrările acestor circuite SI, se face prin programare (similar cu programarea unei memorii PROM).

- matricea sumelor logice - un număr de circuite SAU (atîtea cîte ieşiri are PLA), fiecare cu atîtea intrări cîte ieşiri are matricea produselor.

Conexiunile între MP şi MS sînt de asemenea programabile

- circuitele SAU-EXCLUSIV - atîtea cîte ieşiri are PLA, fiecare cu cîte două intrări. O intrare este legată la ieşirea unui circuit SAU din MS iar cealaltă intrare este programabilă printr-un fuzibil devenind 1 sau 0, în aşa fel încît ieşirea corespunzătoare a MS să apară la ieşirea circuitului SAU-EXCLUSIV, directă sau negată.

- circuitele de autorizare - atîtea cîte ieşiri are PLA, servesc la conectarea în paralel (SAU cablat) a mai multor PLA.

În literatură pentru MP se utilizează denumirea "decodificator incomplet", iar pentru MS: codificator "diluat".

De remarcat faptul că dacă se programează conectarea la un circuit SI din MP atît a unei intrări directe în PLA cît şi a negatei sale, atunci termenul produs corespunzător,este nul.

Rolul esenţial, primar, al PLA este de a genera funcţii directe sau negate, asupra unui anumit număr de intrări, sub

formă de sume de produse logice.

Intre PLA şi o memorie ROM există elemente similare: astfel MP corespunde cu decodificatorul adresei de la memoria ROM, cu deosebirea că la PLA el este incomplet, însă programabil; iar MS corespunde cu matricea de memorie ROM. Blocul SE, la ROM lipseşte şi de aceea, la citirea unui cuvînt din ROM nu se poate obţine opţional şi complementul acestui cuvînt.

La realizarea unor automate complexe, s-ar putea ca utilizarea unei singure capsule PLA să nu fie suficientă. In aceste cazuri pot fi conectate mai multe capsule PLA - în diverse moduri, putîndu-se utiliza şi alte scheme combinaţionale/secvenţiale în vederea realizării scopului dorit.

In fig.3.17 sînt prezentate diverse moduri de interconectare a PLA. In fig.3.17 a,d şi e, se măreşte numărul de combinaţii obţinut la ieşire. In fig.3.17 b şi c, se măreşte numărul de ieşiri, iar în fig.3.17 d şi e, se măreşte numărul de intrări.

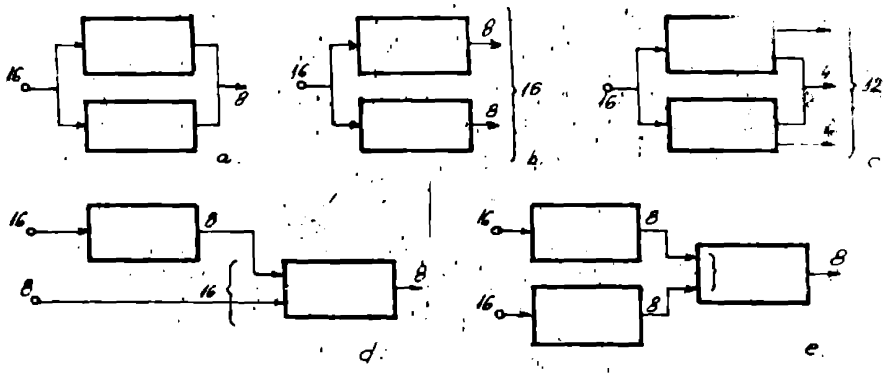


Fig 3.17

In vederea măririi posibilităţilor PLA se utilizează şi scheme combinate, care pe lîngă PLA mai conţin decodificatoare, multiplexoare, memorii ROM, etc., ca în fig.3.18.

Schema din fig.3.18 a, poate fi extinsă prin utilizarea unui decodificator mai mare şi a mai multor PLA. Schema din fig.3.18 b are acelaşi rol ca schema din fig.3.18 a, la fel 3.18 c.

• Schema din fig.3.18 d permite selectarea unei intrări sau a alteia (I_{n-1},I_{n+1}), în funcţie de o altă intrare (I_n).

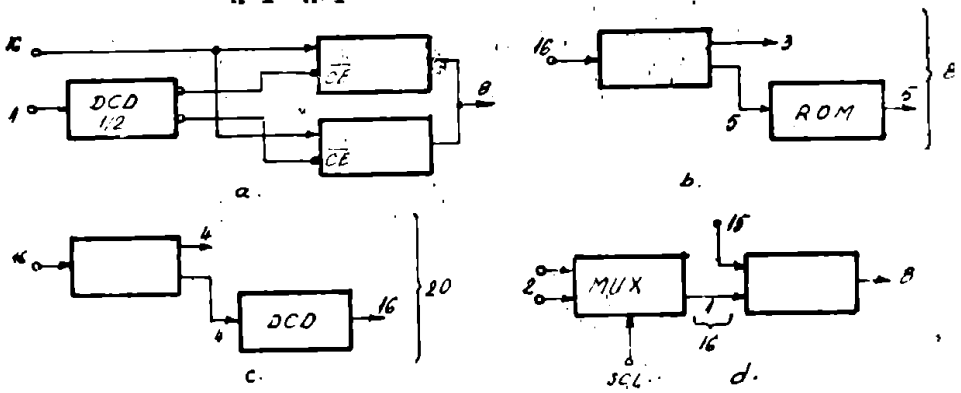


fig. 3.18

Paralingă schemele prezentate, mai pot fi imaginate şi alte moduri de conectare, în funcţie de necesităţile utilizatorului.

In plus, există sub formă de capsule integrate ASS programabile, conţinînd PLA, de tipul FPLS (field programmable logic secuencer), cu o schemă bloc asemănătoare celei din fig.3.15, dar conţinînd în plus un registru pentru sincronizarea funcţiilor de ieşire (cîte un bistabil pentru fiecare ieşire).

3.1.2.5. Automate secvenţiale sincrone cu stări necodificate [H7, P8, P9] (complet decodificate)

Structural aceste automate nu diferă de cele cu stări codificate. Diferenţa dintre ele constă în faptul că în timp ce la cele cu stări codificate logica combinaţională (CSU şi LIE) este complexă, iar registrul de stare RS conţine un număr mic de bistabile, la cele cu stări necodificate, logica combinaţională este foarte simplă, însă RS conţine un număr mult mai mare de bistabili pentru că fiecărei stări sau pas din protocolul de operare îi corespunde cîte un bistabil. Aceste bistabile formează împreună aşanumitul "registru de faze".

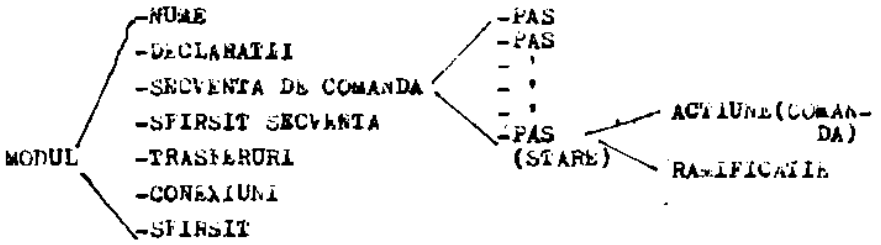
Avantajul esenţial al acestui tip de ASS, este faptul că neexistînd codificare, protocolul de operare se poate transforma foarte simplu de la o structură bidimensională

(tabele, cronograme, organigrame) la o structură monodimensională (program, scris într-un limbaj simbolic).

De asemenea, la acest tip de ASS, prevăzînd o structură arborescentă a MS se pot implementa în mod convenabil protocolurile de operare divergente.

O cale simplă de descriere (pentru înţelegerea funcţionării, sau ca primă etapă de proiectare) pentru aceste automate este utilizarea unui limbaj simbolic, ca de exemplu AHPL derivat din APL .

Limbajul AHPL este realizat spre a servi la descrierea unui întreg SN, din care SD este împărţită în unităţi ce se descriu fiecare separat (blocuri funcţionale), iar SC este împărţită în module (dispozitive de comandă). Fiecare modul, conform sintaxei AHPL, este descris prin următoarele afirmaţii:



Pentru o descriere, proiectare şi analiză concisă ne vom referi în continuare numai la secvenţa de comandă.

Intreaga secvenţă este constituită din paşi, fiecărui pas corespunzîndu-i un bistabil în registrul de stare.

In cadrul unui pas se asigură două funcţiuni:

- funcţiunea de secvenţiere: trecerea la următoarea stare depinde de starea curentă şi de condiţionările exterioare (intrări). Putem avea cazurile: trecere necondiţionată la starea următoare; buclă de aşteptare pe starea curentă; salt necondiţionat la o altă stare (înainte sau înapoi); sfîrşit mort; ramificaţie condiţionată în două sau mai multe direcţii ; divergenţă.

- funcţiunea de comandă: în fiecare pas pot fi generate 0, 1 sau mai multe comenzi în mod necondiţionat sau condiţionat de intrări.

Fiecare stare trebuie să aibă un nume, ex.:1,2,3,..., sau A,B,C,... sau A1,A2,A3,... etc.

O tranziție necondiționată la starea următoare din secvență (pe linia următoare a programului) este subînțeleasă în mod implicit și nu se specifică.

O tranziție necondiționată la o stare oarecare se specifică printr-o săgeată orientată spre dreapta urmată de numele stării de destinație, între paranteze

ex: ⟶(A3) , specifică un salt necondiționat la starea A3.

O buclare condiționată sau o ramificare condiționată se specifică în aceeași manieră:

ex: $\longrightarrow(c_1,c_2,\ldots c_n)/(D_1,D_2,\ldots D_n)$

unde: $c_1,c_2,\ldots c_n$ sînt condițiomările
iar: $D_1,D_2,\ldots D_n$ sînt destinațiile de salt.

In AHPL, introducerea unei valori logice într-un element de memorare se simbolizează printr-o săgeată orientată spre stînga, în stînga fiind notat elementul de memorare iar în dreapta sursa de date (1,0, o funcție logică condiționată sau necondiționată).

ex: x⟵y

Poziționarea unei valori logice "r" pe o linie de semnal "s", se notează astfel:

s = r

De asemenea, în AHPL, scalarii (valori logice singulare) se notează cu litere mici, vectorii (regiștre) cu litere mari, iar tablourile (matrici de memorie) cu caractere grase.

Neinteresîndu-ne destinațiile comenzilor și nici manifestarea efectului lor, nu vom utiliza aceste convenții din considerente de simplificare și pentru a face proiectarea unui DC, mult mai expeditivă.

Astfel, cu aceste considerații un pas de program ce descrie un DC cu stări necodificate poate arăta astfel:

X.⟶(a,a,$\bar{a}$.b,$\bar{a}$.$\bar{b}$)/ (Y,Z,U,X);
cd1=0; cd2=1 ; cd3=a+b.c

ceea ce înseamnă:

- din starea X se poate pleca simultan în două direcții (stările Y și Z dacă a=1) adică are loc o divergență; sau dacă

$\overline{a}.b=1$ se execută o ramificație la starea U; în sfîrșit dacă $\overline{a}.\overline{b}=1$, automatul rămîne în buclă de așteptare în starea X.

- În starea X se generează următoarele comenzi: cd1, cd2 - în mod necondiționat, și cd3 condiționat de relație:

$$a+b.c = 1$$

Evident, acest tip de automat este superior celorlalte tipuri de automate datorită facilităților sale de a asigura foarte simplu funcția de divergență, ce la automatele cu stări codificate este foarte dificil de realizat, iar la cele microprogramate, cu o singură memorie ROM este imposibil de realizat. De asemenea, metoda de proiectare utilizată asigură un înalt grad de formalizare, datorită descrierii liniare a protocolului de operare, eliminînd complet desenele, tabelele, cronogramele, etc. În plus, sînt eliminate toate operațiunile legate de intuiție, tatonare, încercări, reveniri cum există la celelalte tipuri (pentru asigurarea adiacenței stărilor, eliminarea fenomenelor de hazard static sau dinamic datorită intrărilor asincrone ce pot conduce la instalarea unor stări false sau inexistente, generarea unor comenzi parțiale - ieșiri parazite, etc.).

Tehnica de mai sus permite aplicarea unor metode de proiectare automată, cu ajutorul calculatorului cum ar fi:

- utilizarea unui compilator AHPL
- utilizarea unui program de proiectare ce manipulează matrici (tablouri).

În cazul în care se dorește transformarea unei intrări de condiție din nivel asincron în nivel sincron, se utilizează un sincronizator de nivel cu bistabil D, definit prin funcția standard SYN.

Astfel dacă nivelul de intrare asincron este notat cu "a" atunci, acest semnal sincronizat este notat cu:

$$SYN(a)$$

Pentru transformarea unui nivel asincron într-un impuls sincron, cu durata cuprinsă între două impulsuri de tact, semnalului obținut mai sus i se aplică o altă funcție standard, SL și devine:

$$SL\ (SYN(a)),$$

unde funcția SL este obținută printr-o resincronizare și un

produs logic între cele două semnale sincronizate, astfel:

$$SL(SYN(s)) = SYN(s) \cdot \overline{SYN(SYN(s))}$$

In cazul unor variabile dependente, de tip asincron este de asemenea necesară aplicarea funcţiei standard SYN.

De exemplu, dacă într-o anumită stare se declanşează un monostabil "m":

A_i. m←1 ; _ _ _

atunci în altă stare, se utilizează acest monostabil în efectuarea unei ramificaţii sau buclări, astfel:

$A_j \longrightarrow (SYN(m), \overline{SYN(m)}) / (A_K, A_l)$; _ _ _

In sfîrşit, pe baza programului AMPL, proiectarea urmează etape clare şi concise:

- se scriu ecuaţiile de intrare ale bistabililor de tip D din RS, sub formă de sume de produse logice. Există atîţia termeni în ecuaţia logică a unei intrări "i", cîte cazuri de tranziţie directă (necondiţionată) sau condiţionată există spre starea "i". In fiecare termen există ca factori starea din care se face tranziţia, şi condiţionările tranziţiei (dacă există);

- se scriu ecuaţiile comenzilor, tot sub formă de sumă de produse logice, existînd atîţia termeni în sumă cîte apariţii are comanda respectivă în program. In fiecare termen factorii sînt: starea în care apare comanda şi eventual condiţionarea, dacă există.

3.2. Dispozitive de comandă microprogramate

Schemele descrise în paragraful 3.1.2.3. sînt uneori denumite "dispozitive de comandă aproape microprogramate". Diferenţa esenţială între un astfel de dispozitiv şi un DC microprogramat, constă în faptul că acesta din urmă nu conţine în memoria ROM o informaţie, referitoare numai la un protocol de operare ci la mai multe protocoale de operare.

In cazul unui DC microprogramat pentru un CN aceste protocoale de operare se referă la aducerea instrucţiunilor din memorie respectiv la executarea fiecărei instrucţiuni. Astfel

la un calculator care are un set de n instrucţiuni, iar ciclul de instrucţiune se compune din două cicluri de maşină (aducerea instrucţiunii din memoria operativă - executarea instrucţiunii) vor exista n+1 protocoale de operare:

- unul pentru aducerea instrucţiunii ;
- n pentru executarea celor n instrucţiuni

Primul protocol poate fi memorat începînd de la adresa zero, iar celelalte succesiv unul după altul.

Selectarea primului protocol se poate face prin iniţializarea adresei pentru memorie ROM (aducere la zero - o operaţiune simplu de efectuat), iar selectarea unuia din celelalte protocoale specifice ciclurilor de execuţie ale instrucţiunilor se poate face prin convertirea codului instrucţiunii într-un cod de adresă pentru memoria ROM. Acest cod de adresă este adresa de început a protocolului aferent instrucţiunii curente.

In continuare, adresa pentru memoria ROM o vom numi microadresă. Cuvîntul memorat la o anumită microadresă îl vom numi microinstrucţiune, iar un anumit protocol de operare pentru o anumită instrucţiune îl vom numi microprogram. Vom vedea mai tîrziu că va apare şi noţiunea de submicroprogram.

Maurice Wilkes a experimentat pentru prima dată un DC microprogramat la un CN.

Schema de principiu a acestui DC este prezentată în fig.3.19, unde sînt utilizate notaţiile:

ci - codul instrucţiunii
RCI - registrul codului instrucţiunii
DCD - decodificator
A,B,B' - cele 3 secţiuni ale memoriei ROM
μi - cîmpul de comandă A, al microinstrucţiunii, decodificat (comenzi sau microoperaţii).

Cu o secvenţă de μ_i dintr-un microprogram se realizează un ciclu de aducere sau de execuţie al unei instrucţiuni.

MUX - multiplexor de cîmpuri din zonele B sau B' a memoriei ROM. Cu ajutorul lui se selectează una din două microadrese, ca microadrese posibile pentru următorul cuvînt citit din ROM (ramificaţie în microprogram).

C - logică de condiţionare a selecţiei cîmpului de microadresă următoare, din secţiunile B sau B' ale memoriei

ROM. In blocul C sînt cuprinse intrările de condiţie.

△ - întîrziere introdusă pe calea MUX-RCI în scopul de a nu se produce o încărcare prematură a noii microadrese în RCI pînă cînd vechea microadresă mai este încă necesară(pînă la încheierea ciclului de citire a memoriei ROM, decodificarea microinstrucţiunii curente).

Memoria utilizată de Wilkes era o memorie fixă cablată pe inele de ferită,de aici necesitatea impulsului de tact aplicat fie decodificatorului, fie matricii de memorie.

Scopul iniţial al microprogramării, după cum mărturiseşte Wilkes, a fost de a oferi o metodă de proiectare a unui DC mai sistematică şi de aceea mai puţin complicată.

După cum rezultă din fig.3.19, datorită existenţei DCD conectat la ieşirea secţiunii A a memoriei ROM, în această structură poate exista doar o microoperaţie (comandă), pe fiecare microinstrucţiune.

De fapt Wilkes nu utiliza în decodificator,ieşirile secţiunii A a memoriei ROM fiind tocmai liniile de comandă (microoperaţie), astfel că simultan puteau fi generate mai multe comenzi (mai multe microoperaţii pe microinstrucţiune).

- cazul descris în fig.3.19 reprezintă microprogramarea verticală (o microoperaţie pe microinstrucţiune)

- cazul fără decodificator (cîmpul de comandă A, al microinstrucţiunii conţine o informaţie necodificată - adică 1 bit= o comandă) reprezintă microprogramarea verticală.

Microprogramarea verticală necesită un număr mare de cuvinte în memoria ROM, de lungime redusă şi conduce la un timp de execuţie mai lung, întrucît pentru generarea mai multor comenzi trebuiesc citite mai multe cuvinte din ROM.

Microprogramarea orizontală are caracteristicile de mai sus inverse (ROM - capacitate mică, cuvinte de lungime mare, timp de execuţie scurt).

In realitate se folosesc de multe ori, compromisuri între aceste două extreme (aşanumita microprogramare diagonală).

După concepţii mai noi, o microinstrucţiune este de tip vertical dacă controlează o singură resursă hardware (unitatea centrală, memoria operativă, un echipament periferic) ceea determină un salt în microprogram; o microinstrucţiune este de tip orizontal dacă controlează simultan mai multe resurse hardware(şi unitatea centrală şi memoria operativă etc.).

Gradul de codificare al unei microinstrucţiuni poate fi de tipurile:

- fără codificare: fiecare bit specifică o microoperaţie ;
- codificare pe un nivel: microinstrucţiunea este împărţită în cîmpuri, fiecare cîmp fiind conectat la un decodificator. Cîmpurile controlează resurse mutual exclusive.
- codificarea pe 2 sau mai multe nivele: decodificatoarele sînt organizate într-o structură cvasi-piramidală în care ieşirea unui decodificator depinde de alt decodificator.

In fig.3.20 şi 3.21 sînt prezentate două variante principale de realizare a unui DC microprogramat.

In schema din fig.3.20 aducerea microinstrucţiunii din ROM şi executarea microinstrucţiunii sînt operaţiuni seriale; pînă nu se încheie executarea microinstrucţiunii curente,nu se aduce microinstrucţiunea următoare. Cu G.A.U. s-a notat "generatorul adresei următoare".

In schema din fig.3.21 există în plus, un registru de microinstrucţiune. Aici, executarea microinstrucţiunii curente se suprapune cu aducerea microinstrucţiunii următoare. Schema acestui DC se cheamă de tip paralel sau "pipeline", iar registrul microinstrucţiunii se cheamă "registru pipeline".

Datorită faptului că un DC microprogramat are o funcţionare asemănătoare cu a unui CN, a apărut noţiunea de "calculator în calculator", pentru acest tip de DC.

La fel cum instrucţiunile unui calculator pot fi executate prin microprogramare, cu ajutorul microinstrucţiunilor la fel microinstrucţiunile pot fi executate prin "nanoprogramare" cu ajutorul nanoinstrucţiunilor". Avem de fapt de-a face cu microprogramare pe două nivele, şi aşa mai departe .

O altă problemă este ridicată de faptul că orice DC microprogramat conţine pentru comanda sa locală un microsecvenţiator (micro-DC). Acesta la rîndul său, poate fi cablat, sau microprogramat şi aşa mai departe.

In fig.3.22 este prezentat un DC microprogramat, mai simplu, cu o microinstrucţiune la fiecare adresă din ROM,

în fiecare microinstrucțiune fiind conținută o singură microoperație, avînd dispozitivul de comandă locală, cablat.

Codul instrucțiunii din registrul instrucțiunii - RI este transformat de către convertorul de cod - cc într-un cod de adresă. Aceasta este de fapt microadresa primei microinstrucțiuni din microprogramul corespunzător instrucțiunii curente - μAdl. Multiplexorul MUX - μAd lasă să treacă acest cod dacă semnalul de selecție S_{el}=1, apoi μAd - ieșirea se se încarcă paralel în numărătorul de microadrese NA-ROM la comanda de încărcare paralel - IP. La intrarea de adrese a memoriei ROM, IA-ROM apare de fapt în acest moment μAdl. La ieșirea de date ED-ROM, după scurgerea timpului de acces va apare codul primei microinstrucțiuni a instrucțiunii curente,ce se va încărca în RM-ROM, la comanda Pm.

Un cîmp al cuvîntului citit din ROM, adică codul microinstrucțiunii - CμI este trimis la decodificatorul codului microinstrucțiunii. Acesta generează semnalele de comandă $M_1 \ldots M_p$.

Următoarea microadresă poate fi formată prin incrementarea cu 1 a conținutului NA-ROM la comanda +1, sau dacă se execută o microinstrucțiune de salt, cîmpul μAdS va trece prin MUX-μAd dacă S_{el}=1 și apoi se va încărca în NA-ROM la comanda IP, după care ciclul se reia. Un bit din microinstrucțiune poate specifica faptul dacă avem de a face sau nu cu o microinstrucțiune de salt.

O structură microprogramată mai evoluată este prezentată în fig.3.23. Un astfel de tip de DC este utilizat la microprocesoarele bit-alice. Codul instrucțiunii este transformat de o memorie PROM în codul microadresei primei microinstrucțiuni a microprogramului curent. El trece printr-un multiplexor de microadresă, apoi adresează memoria de microprograme ROM. Microinstrucțiunea citită din ROM se va încărca în registrul microinstrucțiunii, de unde, cele patru cîmpuri ale microinstrucțiunii vor fi dirijate astfel:

- un cîmp va fi trimis spre un decodificator, la ieșirea căruia se obțin microoperațiile $M_1 \ldots M_p$.
- alt cîmp, în cazul microinstrucțiunilor ce comandă o ramificare în microprogram (salt), este trimis spre un registru al microadresei de salt, de unde prin multiplexorul de

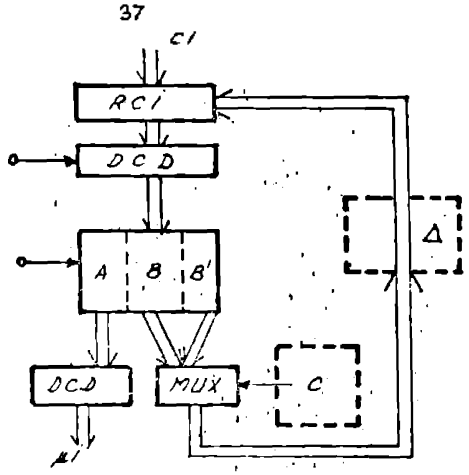


Fig. 3.19.

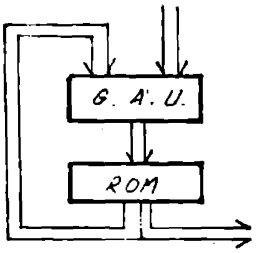


Fig. 3.20.

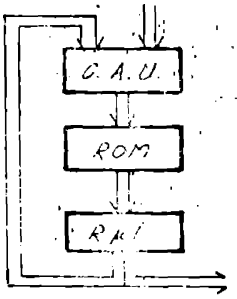


Fig. 3.21

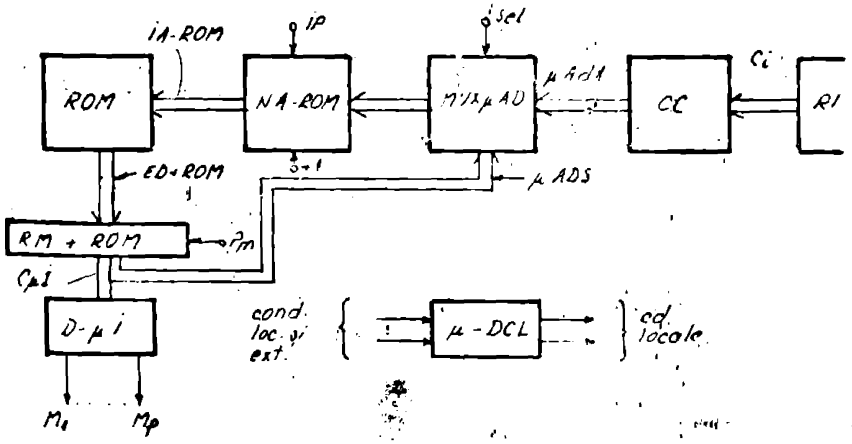


Fig. 3.22.

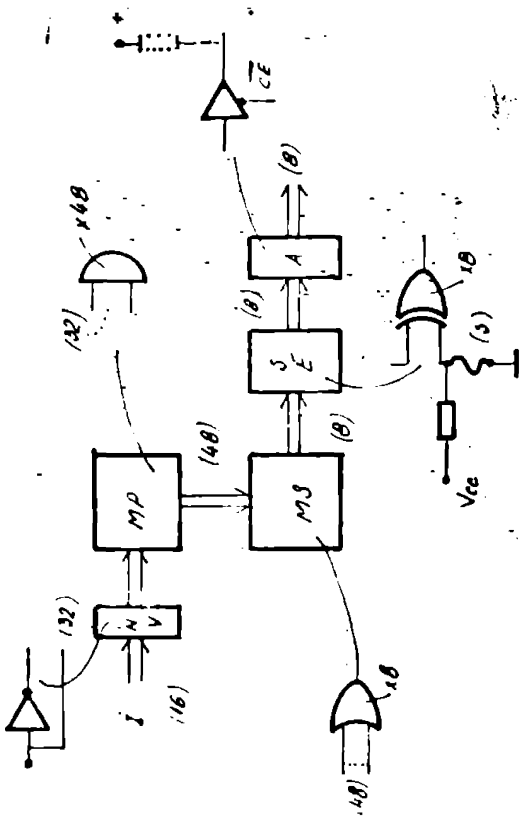


Fig. 3.16

microadresă, va putea adresa memoria de microprograme, etc.

- în fine, cel de-al treilea cîmp, segmentat în două subcîmpuri, serveşte la comanda locală a acestui DC, în felul următor:

- primul subcîmp selectează una din condiţiile externe, cu ajutorul unui multiplexor;

- celălalt subcîmp, combinat cu ieşirea multiplexorului de mai sus, constituie intrarea de adrese a unei alte memorii PROM, pentru comanda locală. La ieşirea acestei memorii PROM, se obţin comenzile locale.

În sfîrşit, dacă microadresa următoare este obţinută prin incrementarea microadresei curente, atu ci ieşirea multiplexorului de microadresă (microadresa curentă) este trimisă la un bloc ce permite incrementarea cu o unitate pentru formarea microadresei microinstrucţiunii următoare, apoi este încărcată într-un registru de microadresă, apre intrarea de adrese a memoriei de microprograme, etc.

În cazul unui salt la submicroprogram, această adresă a microinstrucţiunii următoare, este salvată într-o stivă.

În cazul revenirii din submicroprogram, în microprogra-

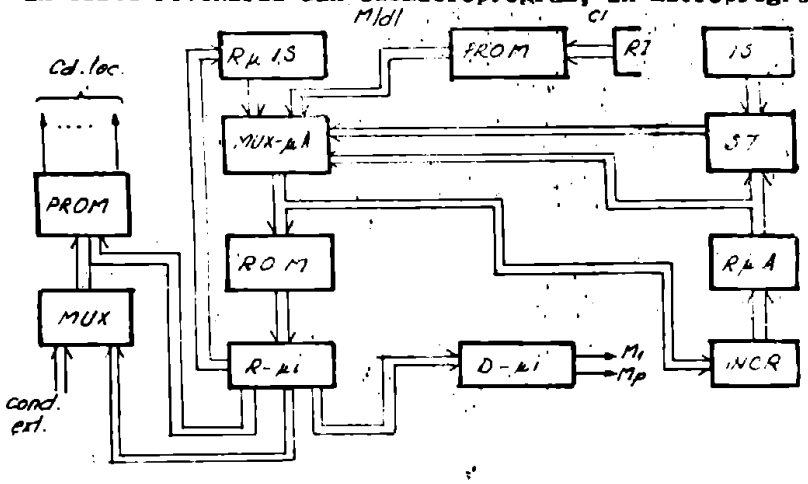


Fig. 3.29.

mul principal (apelant) adresa microinstrucţiunii următoare este regăsită (citită) din stivă.

Adresa vîrfului stivei este specificată de un indicator de stivă.

În concluzie, acestea au fost variantele structurale sau

funcţionale ale DC microprogramate, în rest, alte cazuri nu sînt decît subvariante, legate de modul de organizare al microinstrucţiunilor(verticale, orizontale, diagonale) sau gradul de codificare al lor (necodificate, codificate pe un nivel, pe două nivele, etc.).

Intrucît nu vom analiza în prezenta lucrare decît aspectele hardware ale comenzii, ne vom opri aici cu analizarea DC microprogramate.

3.3. Dispozitive de comandă programate

In cazul în care trebuie realizat un protocol de operare deosebit de complex, iar preţul realizării prin hardware ar fi prohibitiv, se preferă DC programate.

In acest scop, se pot utiliza procesoare universale,procesoare specializate, microprocesoare.

Această variantă este aplicabilă numai în cazurile în care performanţele de viteză de operare cerute nu sînt excesive.

Schema bloc a unui astfel de DC este prezentată în fig. 3.24, unde DC programat conţine unitatea de procesare şi

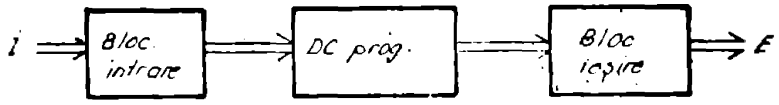


fig 3.24.

memoria cu programele ce determină executarea protocolului de operare cerut. Blocul de intrare preia intrările de condiţionare, le transformă eventual în codul de reprezentare a procesorului, eventual le sincronizează, sau eventual execută o memorare temporară a lor. In cazul celor ce se prezintă sub formă de impuls, pentru a nu se pierde pînă cînd procesorul urmează să le citească în vederea unei testări, conform protocolului de operare. Generarea funcţiilor de ieşire se face prin intermediul blocului de ieşire, prin instrucţiuni de ieşire (scriere) ale procesorului. Blocul de ieşire poate conţine sincronizatoare, bistabile pentru memorarea comenzilor un anumit interval de timp (între două instrucţiuni de ieşire ale procesorului - scriere "1" în bistabil (scriere "0"), monostabile pentru asigurarea de întîrzieri

respectiv formarea în durată a comenzilor, sau alte dispoziti-ve.

Programul rulat în procesor determină momentele de timp - prin subrutine de întîrziere - cînd trebuiesc citite variabilele de intrare, în vederea luării unor decizii, sau momentele cînd trebuiesc generate funcţii de ieşire.

In cazul considerat, procesorul operează (ce intrări/ieşiri) numai asupra unor informaţii numerice de tipul comenzi/stări.

Dacă în paralel cu această funcţiune, procesorul mai execută şi funcţiuni de prelucrări de date numerice/analogice, atunci cele două funcţiuni se pot executa aparent simultan, prin întretesere sau divizarea timpului, de obicei prin intermediul întreruperilor de timp.

Cazul divergenţei comenzii, sau a realizării simultane a mai multor protocoale de operare se rezolvă similar, adică tot prin divizarea timpului. De fapt această problemă, este un caz de emulare simultană a mai multor maşini ţintă(DC-urile ce trebuiesc să funcţioneze aparent simultan) pe o maşină gazdă (DC - programat).

După cum s-a văzut mai înainte, toate cazurile de divergenţă a comenzii sînt dificil de rezolvat, excepţie făcînd cazul DC de tip ASS cu stări necodificate.

Cu anumite dificultăţi, cazuri similare pot fi rezolvate şi cu DC microprogramate. De exemplu, dacă dorim să realizăm pe un DC microprogramat, un grup de protocoale de operare cu execuţie simultană, atunci, fără a utiliza cîte o memorie ROM pentru fiecare protocol de operare, putem utiliza o singură memorie ROM, în care să avem memorate toate protocoalele de operare. In acest caz trebuie utilizat un bloc care să conţină cîte un registru de microadresă următoare pentru fiecare protocol de operare, şi un sistem de comutare a microadresei, după metoda "Round Robin" - comutare în inel, în aşa fel încît să se execute o microinstrucţiune din primul protocol, apoi una din al 2-lea, etc., după care ciclul trebuie reluat. Pentru a asigura o continuitate a timpului real pentru fiecare protocol, microprogramele diverselor protocoale trebuiesc completate cu microinstrucţiuni vide (inoperante), în aşa fel încît toate microprogramele trebuie să fie de aceeaşi lungime,

egală cu microprogramul cel mai lung.

Toate registrele de microadresă următoare trebuie să posede posibilităţi de încrementare, pentru formarea microadresei următoare.

Prin această tehnică s-a obţinut un sistem pe care să-l numim de exemplu "sistem cu microdivizarea timpului".

Continuînd la alt nivel, interior al DC, se pot obţine "sisteme cu manodivizarea timpului".

Revenind la DC programat, acesta, pe lîngă emulare simultană, mai poate asigura şi executarea unor protocoale de comunicaţie între diverse sisteme numerice (ex.mai multe DC programate), prin "macrodivizarea timpului".

Sisteme capabile să asigure divizarea timpului pe mai multe nivele (macro, micro, etc.) sînt sistemele realizate din mai multe microprocesoare bit-slice microprogramabile. In aceste sisteme, însăşi divergenţa comenzii la nivel macro, micro, etc., este relativ uşor realizabilă.

In fig.3.25 este prezentată schema bloc a unui microsistem realizat din module bit-slice. In conformitate cu fig.1.1, aici se evidenţiază foarte clar cele două secţiuni tipice oricărui SN, adică SD şi SC. Referitor la SC (dispozitivul de comandă

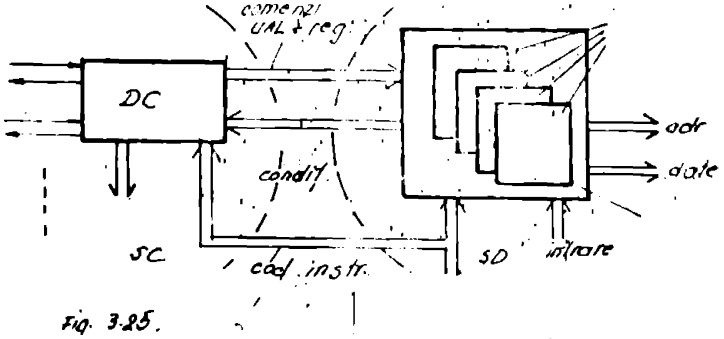


fig. 3.25.

al microsistemului) se remarcă clar intrările de comenzi/stări, ieşirile de comenzi/stări şi intrarea "codul instrucţiunii".

3.4. Criterii de alegere a tipului dispozitivului de comandă

Primul criteriu de alegere a tipului funcţional de DC, este destinaţia sa.

Dacă un DC urmează a fi dispozitivul central de comandă

al unui calculator numeric, atunci el poate fi unul din tipurile:

1 - DC clasic pentru CN (§ 3.1.2.1.)

2 - ASS cu stări necodificate (§ 3.1.2.5.)

3 - DC microprogramat (§ 3.2).

In cazul în care elementele inițiale ale proiectului impun performanțe de viteză deosebite atunci se preferă tipul 1 sau 2. Dacă prețul de cost și flexibilitatea sînt elementele esențiale, se preferă tipul 3. In cazul în care se pune problema divergenței comenzii, se va alege tipul 2, sau o variantă a tipului 3, cu facilități de "microdivizarea timpului", dar numai dacă viteza de operare este neesențială.

In SN de tip asincron (ex.memorii cu ferite sau semiconductoare) se preferă DC realizate cu monostabile(§ 3.1.1.1) sau cele cu linii de întîrziere (§ 3.1.1.2) ultimele fiind mai precise și permițînd realizarea unor protocoale de operare foarte rapide. DC cu linii de întîrziere se utilizează și ca DC sincrone, în sistemele în care cerințele de viteză sînt excesive (ex.bază de timp pentru blocuri de memorie MOS).

In cazul unui protocol de operare deosebit de complex, sau în cazul mai multor protocoale de operare cu execuție în paralel, dar numai dacă cerințele de viteză nu sînt deosebit de mare, se utilizează DC programate (§ 3.3). Dacă și cerințele de viteză sînt mari, atunci se utilizează DC combinate cu memorii ROM, PLA (§ 3.1.2.3, 3.1.2.4). La cerințe de viteză excesive se utilizează DC de tip ASS cu stări necodificate, cu facilități de divergență (§ 3.1.2.5).

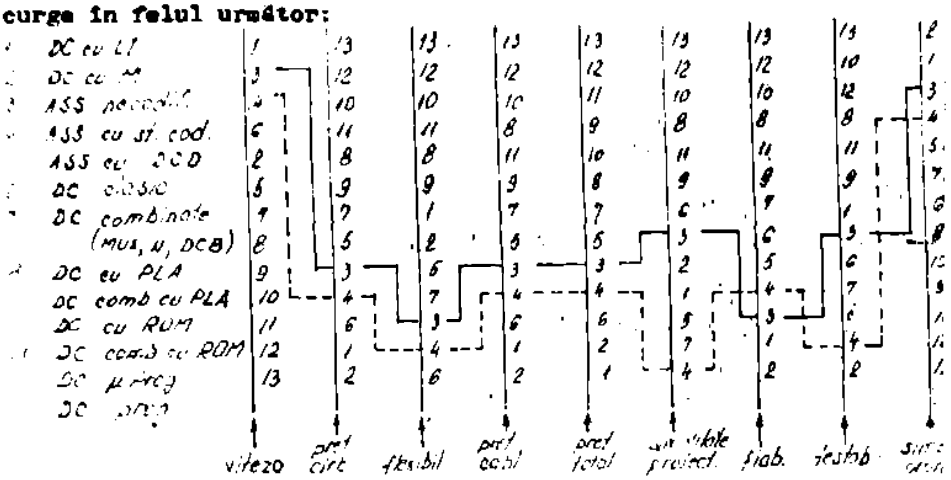
Dacă protocolul de operare este segmentat iar segmentele se cuplează prin bucle de așteptare, sau se utilizează tehnici de cuplare simulare cuplării unor subprograme la un program principal, atunci pe lîngă DC programate, se pot utiliza ASS cu stări codificate sau necodificate, ce permit ușor segmentarea și cuplarea.

Cînd se pune problema unei proiectări ușoare, și a unei testabilități ridicate, dintre ASS, se alege tipul cu stări necodificate.

In cazurile în care fiabilitatea sistemului este un factor esențial se preferă schemele ce conțin cît mai puține componente și cît mai puține cableje ca de exemplu: DC

programat, DC microprogramat, automate cu memorii ROM, cu retele logice programabile.

In fig.3.26 este prezentată o diagramă pe baza căreia proiectantul unui DC va putea alege tipul funcţionalo-constructiv cel mai adecvat în funcţie de cerinţe. Alegerea decurge în felul următor:



- pe linia verticală corespunzătoare cerinţei prioritare de performanţă a DC (viteză de operare, flexibilitate,etc.) se va alege numărul din vîrful listei.

Se vor uni cu o linie întreruptă, toate aparițiile acestui număr pe celelalte linii verticale. Dacă curba obţinută - cu excepţia maximului - se află la un nivel scăzut, se procedează similar cu al 2-lea element de pe prima verticală considerată. Dacă linia întreruptă ce va rezulta se află la un nivel mult mai ridicat decît prima linie, atunci numărul găsit (al 2-lea de pe verticala considerată) se caută în lista tipurilor de DC, şi în felul acesta s-a stabilit ce tip este cel mai adecvat aplicaţiei considerate. Prin metoda considerată, din aproape în aproape, se poate determina un compromis optim în cazul soluţiei alese.

4. SISTEME DE DISPOZITIVE DE COMANDĂ, DIALOGUL ÎNTRE DC [A3, C5, H7, C18].

În capitolele anterioare s-au analizat toate variantele constructivo-funcţionale de dispozitive de comandă, plecînd de la faptul că pentru un SN cu SD cunoscută, trebuie pro-

iectată SC. S-a mai considerat că SD nu conţine elemente de comandă locală, iar SC conţine un singur DC. In realitate, chiar în cadrul unui singur SN, SD poate conţine elemente de comandă locală sau o întreagă structură de comandă pe mai multe nivele (cu alte cuvinte, o întreagă secţiune de comandă în secţiunea de date).

In plus, în SC, putem avea nu un singur DC ci mai multe; pot exista de asemenea şi blocuri specifice prelucrării de date.

In cazul reţelelor de sisteme numerice fiecare cu SD şi SC proprii, în care acestea trebuie să conlucreze sau să execute dialog între ele, evident comanda este multiplă - pentru că fiecare SN are cel puţin o SC - iar în cadrul fiecărei SC putem avea o comandă structurată pe nivele: comanda centrală, comenzi locale. Dacă două SN execută dialog atunci comanda dialogului este controlată de ambele SC ale celor două SN.

Au fost deja prezentate:

- comanda primară (fig.1.1) - un singur DC/nivel ;
- comanda pe două nivele:MASTER şi SLAVE (fig.1.2).

Nivelul MASTER conţine un DC, nivelul SLAVE - un DC.

Generalizînd schema comenzii pe două nivele, se ajunge la schema din fig.4.1, în care există un DC - MASTER şi mai multe DC - SLAVE.

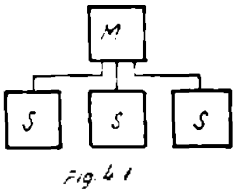


Fig. 4.1

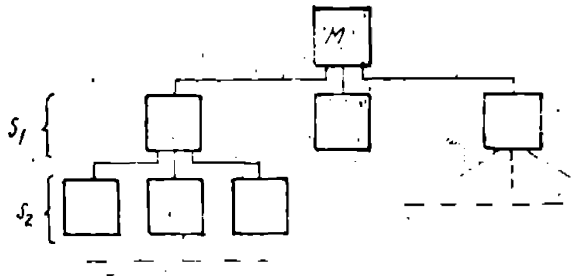


Fig. 4.2

In acest caz, dacă DC-MASTER nu conţine nici un punct de divergenţă (comandă paralelă) el nu poate efectua dialog decît cu un singur DC-SLAVE la un moment dat, cu cel solicitat de MASTER (M) sau cu cel mai prioritar SLAVE (S) din cele care au avut dialog.

- comanda pe trei sau mai multe nivele cu respectarea ierarhiei. In fig.4.2 - exprepolare a fig.4.1 - există un DC-MASTER, un grup de DC-SLAVE 1, un grup de DC-SLAVE 2,

etc. Dialogul se poate efectua numai între M-S1, sau S1-S2, sau S2-S3, etc. Exemplu tipic ar fi: M - calculator central, S1 - calculatoare satelit, S2 - periferice conectate doar la calculatoarele satelit. Fiecare dintre M, S1, S2, au DC propriu.

- comanda pe trei sau mai multe nivele fără respectarea ierarhiei. In fig.4.3, asemănătoare fig.4.2, ierarhizarea este asemănătoare, dar dialogul este permis prin omiterea nivelelor intermediare. In plus, pot exista două sau mai multe blocuri pe nivelul 0 - $M_1, M_2, \ldots$

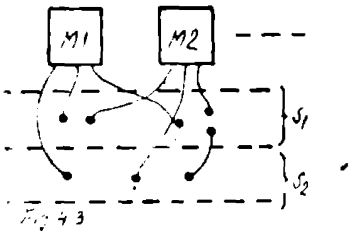


Fig. 4.3

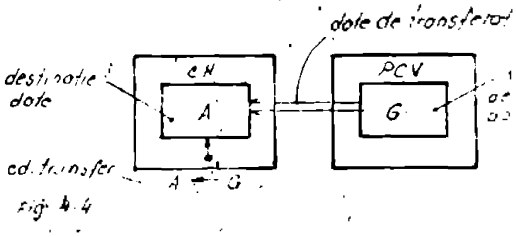


Fig. 4.4

Datorită existenţei mai multor blocuri M, pot exista două sau mai multe dialoguri simultane cu condiţia să nu apară concurenţă, adică două dialoguri simultane să nu implice aceleaşi resurse, dacă aceste resurse nu au puncte de divergenţă în vederea executării unor operaţii paralele.

- comanda întreţesută cu ierarhie variabilă sau flexibilă (sisteme interactive). Un astfel de tip de comandă este permis de structura prezentată în fig.1.3 (a se revedea şi explicaţiile de la această figură !).

Indiferent de structura analizată şi de ierarhizarea acestei structuri, se pun două probleme referitoare la comandă:

- comanda locală - analizată în detaliu în cap.2 şi 3.

- comanda globală - comandă ce se referă la minimum două DC implicate în dialog.

Prin extrapolare, problema unui dialog se poate extinde la mai multe dialoguri simultane, sau un dialog în care sînt implicate mai multe DC momentan.

In toate cazurile ce le putem considera, fie că dorim să conectăm între ele două automate secvenţiale sincrone; sau două calculatoare, sau un echipament periferic la un calculator, problemele care se pun sînt similare, ceea ce diferă este doar complexitatea dialogului.

Vom analiza în continuare două cazuri situate la cele două extremităţi ale complexităţii dialogului.

Considerăm mai întîi cazul cel mai simplu şi anume: un SN implicat în dialog este un CN iar celălalt este un panou de comandă şi vizualizare prevăzut cu un registru de comutatoare (generator de cuvinte). La un moment dat, printr-o instrucţiune, calculatorul transferă conţinutul generatorului de cuvinte al panoului de comandă, în acumulatorul său intern.

Acest dialog simplu presupune următoarele:

- în momentul apariţiei instrucţiunii de transfer datele trebuie să fie prezente şi stabile în sursa de date.
- trebuie să existe o conexiune directă, monodirecţională între sursa de date şi destinaţie.
- blocul de destinaţie (acumulatorul) trebuie să fie disponibil (liber) pentru preluarea datelor.
- în momentul executării acestei instrucţiuni, DC al CN trebuie să genereze un semnal de comandă ce iniţiază transferul, A←G, ca în fig.4.4.

In acest caz simplu de comunicaţie între două SN, nici măcar nu este vorba de un dialog în adevăratul sens al cuvîntului.

Un caz ceva mai complex, este cuplarea a două automate secvenţiale sincrone. In acest caz, dialogul este la nivelul lansare comandă - verificare atare, pentru ambele automate, fără a implica un transfer de date.

Spre limita superioară a complexităţii, comunicaţie între două SN dintr-o reţea de SN, are loc în felul următor: presupunem mai multe SN notate $SN_1, SN_2, \ldots, SN_i, SN_j, \ldots SN_m$ conectate între ele prin magistrale de date, adrese, stări şi comenzi. Oricare dintre ele poate iniţia un transfer cu orice alt SN din cele rămase.

Vom presupune că SN_i va iniţia transferul unor date sau în general dialogul cu SN_j, indiferent care din ele este sursă şi care destinaţie a datelor, pentru că ne interesează de fapt care este dialogul la nivelul: comenzi/stări şi nu transferul propriuzis de date. Considerăm că SN_i şi SN_j sînt asincrone unul în raport cu celălalt.

Cazul cu mai multe SN de destinaţie, nu va fi analizat, pentru că poate fi dedus prin extrapolare.

Comunicaţia sau dialogul între SN_i şi SN_j va decurge astfel:

- SN_i lansează o comandă sau un semnal ce iniţiază dialogul.

- se memorează această comandă pentru a nu se pierde, în cazul că SN_j nu este capabil să o observe imediat, datorită sarcinilor sale curente.

- în paralel cu comanda, SN_i generează şi adresa sau codul de selecţie (cuvîntul de identificare) propriu numai lui SN_j.

- fiecare SN are un decodificator (segment de decodificator) capabil să detecteze momentul cînd adresa lui este prezentă pe magistrala de adrese simultan cu existenţa comenzii de transfer. În cazul nostru, adresa este identificată de către SN_j. Aceasta poate avea şi un sincronizator, necesar pentru sincronizarea comenzii cu tactul său propriu, dacă SN_i şi SN_j sînt asincrone.

- SN_j lansează un răspuns prin care anunţă SN_i că a recepţionat comanda, fiind chiar el dispozitivul solicitat.

- în paralel cu acest răspuns, SN_j prezintă pe o magistrală de stare, cuvîntul său propriu de stare (conţinutul unui registru sau grup de bistabile).

Cuvîntul de stare diferă de la SN la SN, uneori fiind doar de un bit care indică faptul că SN considerat este apt sau inapt, sau chiar poate lipsi, dacă SN este întotdeauna apt pentru comunicaţie (cazul generatorului de cuvinte).

Dacă SN_i care lansează comanda sau cererea de dialog, plasează acest semnal pe o magistrală bidirecţională de 1 bit, la care sînt conectate şi celelalte SN, atunci SN_i va trebui să mai plaseze pe o magistrală de comenzi codul tipului de dialog solicitat (interogare stare, intrare date, ieşire date) şi eventual codul propriu lui SN_i. În cazul dialogului iniţiat prin întrerupere, cererea de dialog este plasată pentru fiecare SN pe o linie reparată astfel că nu mai este necesar codul SN_i.

- revenind la momentul cînd SN_j a acceptat comanda şi şi-a plasat cuvîntul de stare pe magistrala de comenzi-stări, SN_i va reacţiona la primirea stării din SN_j printr-un semnal ce-l va plasa pe linia de acceptare. Dacă SN_j nu este apt pentru dialog, atunci SN_i poate trece la alte

operaţiuni locale, proprii, sau dacă cererea de dialog este critică (număr mare de solicitări rămase fără răspuns, sau timp prea lung scurs din momentul cererii) atunci SN_i iniţiază din nou dialogul.

- din acest moment, sursa de date (SN_i sau SN_j) pregăteşte datele de transferat şi anunţă printr-un semnal că datele sînt disponibile (valide).

- SN care va prelua datele, observă acest semnal, preia datele şi anunţă restul de SN că este ocupat, poziţionînd pe 1 o linie de semnal.

- cînd datele sînt recepţionate complet, SN receptor anunţă aceasta printr-un alt semnal, care arată că dialogul s-a încheiat;

- dacă receptorul va mai fi ocupat în continuare cu prelucrarea acestor date, atunci pe linia care anunţă că este ocupat vom avea în continuare valoarea 1, pînă cînd receptorul de date este complet disponibil pentru un nou dialog. De exemplu dacă SN receptor este un perforator de bandă, atunci după ce el a recepţionat cuvîntul care trebuie perforat, în registrul său tampon, va pierde încă un timp pentru procesele mecanice lente legate de perforare propriu-zisă şi avansul benzii de hîrtie.

- orice SN angajat într-un dialog, trebuie să fie capabil să răspundă unei cereri de citire a stării sale. Deci tot ceea ce s-a descris pînă acum poate fi împărţit în două cazuri: dialog-stare, dialog date. Dacă dialogul-stare este de natură să permită dialogul-date, atunci va urma acest dialog.

In majoritatea cazurilor, dialogul dintre SN nu este de această complexitate.

Pentru detalierea dialogului spre limita maximă a complexităţii, în fig.4.5 sînt prezentate SN_i şi SN_j cu conexiunile dintre ele, iar în fig.4.6, o diagramă ce ilustrează fazele sau etapele dialogului.

Pentru cazurile particulare de dialog (prin întrerupere, acces direct la memorie) sînt prezentate comentarii în fig. 4.6.

Cazul tratat mai sus, presupune existenţa unui singur dialog momentan (sau cu alte cuvinte, dialoguri divizate sau multiplexate în timp).

Pentru două dialoguri simultane, toate magistralele din fig.4.5 trebuiesc dublate, pentru 3 dialoguri simultane, triplate, etc.

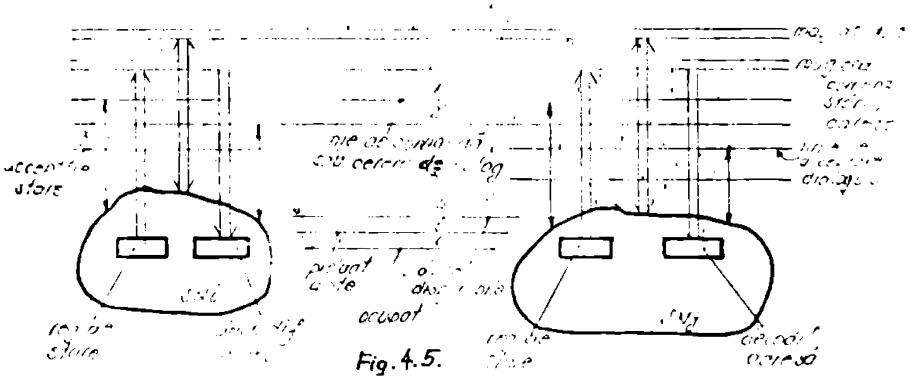


Fig. 4.5.

Pentru cuplarea mai multor SN, se va alege o soluţie de compromis între complexitatea sistemului de magistrale, şi numărul de dialoguri simultane dorite.

Dacă la un moment dat, diverse SN solicită dialog prin semnalele de iniţiere $i_1,\ldots,i_n$, atunci va fi necesară o schemă de priorităţi care va stabili ce dialog va avea loc din cele solicitate, permiţînd poziţionarea pe magistrala A - adresant şi pe magistrala I - iniţiator, codurile SN angajate în dialogul acceptat.

S-au considerat distinct magistralele A şi I pentru simplificarea înţelegerii, ele fiind de fapt componente ale magistralei de comenzi-stări-adrese. În fig.4.7 este prezentat acest mecanism valabil în cazul unui singur dialog momentan (arbitru de magistrale).

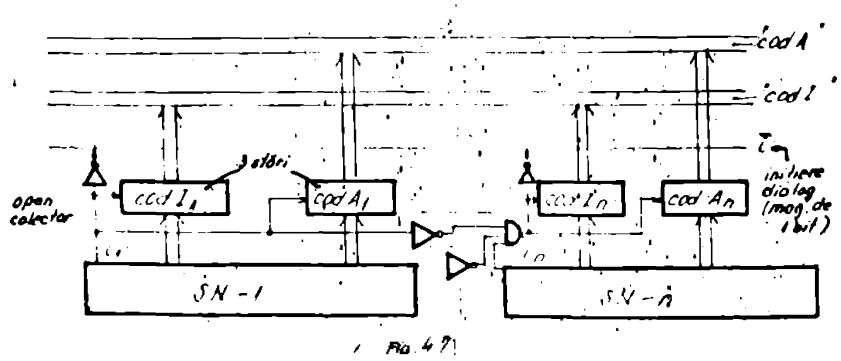


Fig. 4.7

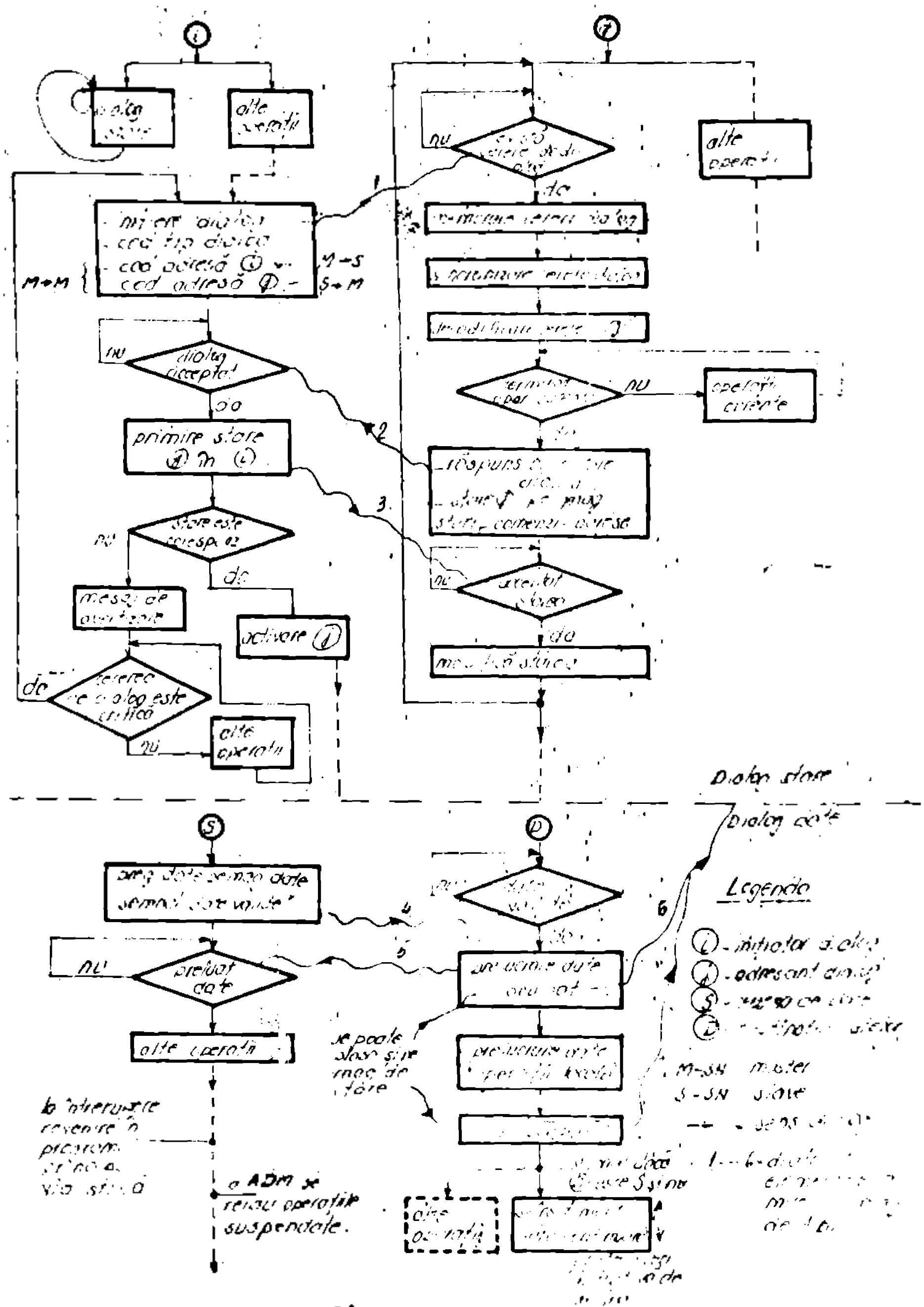

alte operatii
primire stare
activare
Dialog stare
Dialog date
Legenda
alte operatii
a ADM se reiau operatiile suspendate.

Fig. 46.

Cele 6 magistrale de 1 bit din fig.4.5 pot fi reduse ca număr, sau parte din ele pot fi incluse pe magistrala de comenzi-stări-adrese.

Astfel liniile:acceptare dialog, acceptare stare şi preluat date pot fi reduse la o singură magistrală bidirecţională de răspuns la orice solicitare.

Sistemele multimicroprocesor reprezintă un caz particular de cuplare a mai multor SN.

După modul de interconectare, un sistem multimicroprocesor poate fi de tipul :

- ierarhizat (un μP-master şi celelalte μP-slave)
- cu funcţionare în paralel
- în inel
- cu funcţionare în comutaţie
- interactiv (cu configuraţie flexibilă, variabilă în funcţie de solicitările externe).

Se va remarca faptul că în cazul unui dialog complex de tipul celui prezentat în fig.4.6, în timp ce un SN este angajat într-un transfer de date, el trebuie să fie capabil să răspundă unei cereri de dialog-stare. Astfel apar două operaţii paralele sau simultane (divergenţă pe două ramuri)

5. ADAPTAREA UNOR METODE DE PROIECTARE ÎN VEDEREA PROIECTĂRII AUTOMATE

După cum s-a văzut în capitolele anterioare, majoritatea metodelor de proiectare a DC au la bază, exceptînd etapele intermediare, fie organigramele de stări[C/8], fie tabelele[R5], fie programele scrise într-un limbaj simbolic [H7](ex.:AHPL). În prezenta lucrare se utilizează toate cele trei metode, în ultimul caz, limbajul AHPL fiind restrîns şi condensat în varianta AHPL-X, specifică numai strict comenzii (luarea în considerare a intrărilor, asigurarea funcţiei de secvenţiere, generarea funcţiilor de ieşire). Indiferent ce metodă de bază se utilizează, aceasta se poate rezuma în final la un set de tabele, ce pot fi prelucrate în mod automat, cu ajutorul calculatorului. Revenind la programele scrise în AHPL sau varianta AHPL-X, acestea pot fi prelucrate cu creionul şi hîrtia (prin căutarea tranziţiilor sau a ieşirilor), sau utilizînd un compilator. Prima metodă este

utilizată în lucrarea[C9], a doua depăşeşte cadrul lucrării.

Revenind la metodele tabelare, foarte des utilizate, pînă în prezent prelucrate manual, tot prin metode de căutare, vom încerca să elaborăm o metodologie de întocmire a acestor tabele şi un algoritm de prelucrare a lor, în vederea proiectării DC cu ajutorul calculatorului.

In acest sens, vom pleca de la enumerarea şi analizarea tipurilor de tabele utilizate pînă în prezent.

- primul tip de tabele considerate [R5] , aplicate la descrierea DC al unui CN este format în felul următor: pentru fiecare fază a fiecărui instrucţiuni sau comenzi manuale se întocmeşte un tabel. In fiecare tabel există atîtea coloane cîte impulsuri ale bazei de timp conţine faza considerată (ciclu de maşină). In fiecare coloană sînt enumerate comenzile - în notaţii mnemonice - ce se generează în starea corespunzătoare coloanei respective. Comenzile condiţionate sînt date descriptiv, ca de exemplu:

dacă X=1, atunci $i_j \rightarrow Z$,
dacă Y=1, atunci $i_j \rightarrow Y$.

Tranziţiile directe dintr-o stare în următoarea (în ordine) nu sînt specificate.

Tranziţiile condiţionate sînt specificate tot descriptiv
dacă Q=1, atunci treci la i_K,
dacă $\overline{Q}$=1, atunci treci la i_ℓ

In acest tip de tabel, baza de timp are un număr fix de stări.

Semnificaţiile notaţiilor sînt următoarele:
X,Q - condiţionări
Z,Y - funcţii de ieşire (comenzi)
i_j, i_K, i_l - stări ale bazei de timp.

După tratarea funcţiilor de ieşire şi a secvenţionării, în porţiunea inferioară, aceste tabele mai conţin alte porţiuni descriptive (explicaţii, specificaţii asupra fronturilor active, asupra întîrzierilor introduse intenţionat pe ieşirile bazei de timp, etc.).

Datorită elementelor descriptive, acest tip de tabele nu se pretează fără transformări de structură la proiectarea automată.

- alt tip de tabele[R6], aplicat tot la descrierea DC a

unui CN, conţine un tabel pentru toate instrucţiunile (cîte una pe o linie) desfăşurate pe cicluri de maşină (faze) şi stări ale bazei de timp, numărul de stări şi numărul de cicluri maşină fiind variabil de la instrucţiune la instrucţiune. Tabelul conţine pentru fiecare instrucţiune sau linie, următoarele: mnemonicul instrucţiunii, codul instrucţiunii, iar apoi "n" coloane corespunzătoare numărului maxim de cicluri de maşină (n), şi fiecare din aceste coloane împărţită în atîtea subcoloane cîte stări maxime posibile există în ciclul maşină considerat, comenzile ce apar. Coloanele şi subcoloanele neutilizate rămîn vide (nu se completează).

In acest tip de tabel nu se specifică explicit comenzile ci sînt descrise simbolic, operaţiuni asupra datelor, ca de exemplu:

PC = PC + 1 (incrementare numărător de adrese - Programm Counter)

sau: (A)⟶ALU , ROTATE _____ALU⟶A,CY

(transferarea conţinutului acumulatorului în unitatea aritmetico-logică, efectuarea unei rotiri, transferarea rezultatului în acumulator şi în bistabilul de transport CY - carry).

In ceea ce priveşte partea de secvenţiere, este de remarcat faptul că la acest tip de tabele există numai tranziţii directe din stare în stare, cu omiterea eventuală a cîtorva din ultimele stări ale ciclului de maşină curent, trecerea la noul ciclu de maşină făcîndu-se direct, din ciclu în ciclu-atîtea cicluri cît sînt necesare. In diagrama stărilor nu pot exista decît bucle de aşteptare pe o stare, fără salt înapoi, fără ramificaţie şi fără puncte de divergenţă.

De remarcat faptul că funcţiile de comandă nu apar explicit, cît trebuiesc deduse din transferurile de date.

In plus, funcţiile condiţionate sînt date descriptiv, ca de exemplu:

```
JUDGE       CONDITION
IF  TRUE, SP = SP-1
```

(dacă condiţia pusă este adevărată, se decrementează indicatorul de stivă).

Pe lîngă limitările de structură menţionate, partea descriptivă existentă în acest tip de tabel, nu îl impune în vederea proiectării automate.

- ultimul tip de tabele [H7] analizate, pleacă de la pro-

gramul AHPL ce specifică protocolul de operare, pentru un ASS cu stări necodificate (şi pentru secţiunea de date, dar care aici nu interesează).

Un astfel de tabel pentru secţiunea de comandă conţine atîtea linii cîţi paşi există în secvenţă, pentru fiecare pas specificîndu-se: numele/numărul pasului, ecuaţia D pentru bistabilul aferent pasului, numele bistabilului aferent pasului, stările următoare stării curente şi condiţionările lor, dacă există, şi în final funcţiile de ieşire condiţionate sau nu, din pasul respectiv. Prelucrarea acestui tip de tabele presupune existenţa unui compilator AHPL. Este posibilă şi o compilare manuală, prin inspectarea tabelului. Considerăm că întocmirea acestui tabel pe baza programului este greoaie şi conţine prea multe date spre a fi util în cazul unor protocoale de operare complexe. Ca elemente pozitive menţionăm:eliminarea elementelor descriptive, posibilitatea simplă de a introduce puncte de divergenţă, uşurinţă în înţelegerea protocolului de operare pe baza tabelului.

In continuare vom elabora o metodă tabelară care să întrunească elementele pozitive ale celor trei tipuri de tabele, şi să elimine aspectele negative semnalate mai sus .

Ca punct de plecare, vom alege organigrama pentru descrierea protocolului de operare, datorită posibilităţilor ce le oferă, iar ca structură hardware vom considera un automat secvenţial, sincron cu stări necodificate.

Dacă stările automatului sînt notate numeric (1,2,3,...) atunci ele pot intra în tabelul ce va urma, direct în această formă. Dacă nu, atunci va trebui întocmit un dicţionar în care numele fiecărei stări este reprezentat printr-un număr. Astfel de dicţionare se vor întocmi pentru intrările de condiţionare (sau pentru combinaţiile de intrări) şi pentru funcţiile de ieşire. Pe baza acestor trei (două) dicţionare şi pe baza organigramei protocolului de operare se va întocmi tabelul.Schematic acest tabel ar putea fi reprezentat ca un tablou tridimensional, ca în fig.5.1.

In această figură, s-a rezervat un spaţiu de forma unui paralelipiped cu 3 elemente pe orizontală, $2^3+1(9)$ pe verticală şi 2^6+1 (257) în adîncime.

Aceasta în vederea cazului în care pentru proiectarea automată se alege un limbaj de programare în care declararea di-

mensiunilor de tablou se face static (ex.FORTRAN), adică necunoscînd de la început pentru fiecare aplicaţie, ce dimensiuni are tabloul, se dau declaraţii de dimensionare acoperitoare.

Dimensiunea cea mai mare este explorată sau mercată cu indicele i, următoarea cu j şi ultima cu k.

Deci:

$$i = 1 - 2^8 + 1$$
$$j = 1 - 2^3 + 1$$
$$k = 1, 2, 3.$$

i - fiind numărul curent al stării;

j - fiind numărul curent al condiţionării dintr-o anumită stare.

k - fiind un indice ce specifică ramificaţia sau funcţia de ieşire, condiţionate sau nu, de intrările de condiţii. Astfel:

k = 1 - indică condiţionarea curentă ;
k = 2 - ramificaţia în secvenţă(starea următoare);
k = 3 - funcţia de ieşire.

Tabloul de intrare, ce descrie protocolul de operare va fi notat cu TI (i,j,k)

Dacă Ti(i,j,1)=0, nu există nici o condiţionare pentru:

Ti(i,j,2)
şi Ti(i,j,3)

Dacă Ti(i,j,2) = 0, atunci nu există continuare în secvenţă (sfîrşit mort).

Dacă Ti(i,j,3) = 0 atunci nu există funcţie de ieşire în starea i, varianta j.

Pentru ca Ti să nu fie completat cu zerouri pentru toate locaţiile libere, atunci cînd într-o anumită stare $j_M < 8$, vom considera următorul element:

$$Ti(i, j_M+1, 1) = -1$$

unde j_M - este numărul de ordine al ultimei variante considerate pentru condiţii, în starea i.

Vom remarca că se consideră într-o anumită stare, pe lîngă condiţionările obişnuite şi o condiţionare nulă(adică tranziţie sau funcţie de ieşire necondiţionată), cînd:

$$Ti(i,j,1) = 0$$

După elementul $Ti(i_M, j_M, k_M)$, din aceleași considerente, vom lua

$$Ti(i_M+1,1,1) = -2$$

unde i_M - este numărul de ordine al ultimei stări.

Dacă: Ti(i,8,3) este ultimul element considerat în starea i, atunci:

$$Ti(i,9,1) = -1$$

La fel, dacă:

Ti(256,j_M,3) este ultimul element (j_M), considerat în ultima stare posibilă (256), atunci va urma perechea de elemente:

$$Ti(i_M, j_M+1, 1) = -1$$

și $$Ti(i_M+1,1,1) = -2$$

Pe baza organigramei protocolului de operare, tabelul Ti se completează astfel:

- pentru starea 1, varianta de condiționare 1 se vor nota elemente:

$$Ti(1,1,1),\ Ti(1,1,2),\ Ti(1,1,3)$$

unde:

Ti(1,1,1) - reprezintă prima condiționare și este de fapt numărul de ordine al condiționării curente din dicționarul de condiționări.

Ti(1,1,2) - reprezintă numărul de ordine al stării următoare stării 1, în condiționarea 1 și va avea valoarea zero dacă există sfîrșit mort.

Ti(1,1,3) - reprezintă numărul de ordine al funcției de ieșire generată în starea 1 în condiționarea 1.

În felul acesta s-a completat prima linie din tabelul Ti.

Următoarea linie:

$$Ti(i,2,1),\ Ti(i,2,2),\ Ti(i,2,3)$$

se completează similar, etc.

Pe linia j_M+1 nu există decît un element:

$$Ti(i,j_M+1,1) = -1$$

Se procedează similar pentru fiecare stare încheindu-se în final cu elementul:

$$Ti(i_M+1,1,1) = -2$$

Astfel s-a întocmit tabelul de intrare Ti.

Avantajele nete ale utilizării acestui tip de tabel pentru descrierea protocolului de operare sînt următoarele:

- tabelul este de tip pur numeric, deci foarte ușor de manipulat în cazul prelucrării sale cu ajutorul calculatorului

- nu există nici o parte descriptivă în tabel, totul este riguros și concis.

- cele 3 dicționare anexe tabelului, pot fi eliminate dacă intrările, stările și ieșirile sînt notate în ordine, ca de exemplu:

$S_1, S_2, \ldots, S_n$ - pentru stări

$C_1, C_2, \ldots, C_m$ - pentru intrări (condiții)

$F_1, F_2, \ldots, F_p$ - pentru funcțiile de ieșire.

Marcarea acestora în tabel este simplă, întrucît se înlătură prefixul (S,C,F) rămînînd doar indicele (1,2,...n sau m sau p).

- permite ramificații de toate tipurile (salt înainte, salt înapoi, buclă de așteptare, ramificații condiționate în mai multe direcții (aici ≤ 8).

- permite implementarea comenzii divergente, condiționată sau necondiționată.

- permite implementarea funcției de convergență între două sau mai multe ramuri paralele.

- nu necesită compilator ci pur și simplu un program utilizator.

- permite existența oricîtor puncte de SPINSIT MOST.

- tabelul se poate întocmi pe baza unei descrieri primare a protocolului de operare (organigrama de stări).

- în urma prelucrării tabelului Ti se obțin două tabele: TS - care descrie funcția de secventiere a DC sub forma ecuațiilor de intrare date tabelar, a intrărilor D a bistabililor registrului de stare și TF - tabelul funcțiilor de ieșire, care ca și TS este dat sub formă de sumă de produse logice tabelate, lipsind doar operatorii:

\+ - sumă logică

. - produs logic

TS și TF pot fi obținute și ca ecuații finale dacă se utilizează operatorii . și + .

În continuare vom analiza modul cum se întocmește tabelul

Ti, vom arăta cum se prelucrează acest tabel şi cum se obţin tabelele finale cu soluţia problemei, TS şi TF.

De remarcat faptul că pe baza TS şi TF se face sinteza schemei DC, iar funcţiile de convergenţă se stagează ulterior, ca blocuri funcţionale. De exemplu, dacă X şi Y sînt ultimele stări ale unor ramuri paralele în DC atunci ieşirile bistabililor de stare corespunzători stărilor X şi Y se introduc într-o schemă de convergenţă CV (X,Y) iar ieşirea schemei de convergenţă constituie intrarea D a bistabilului corespunzător primei stări din trunchiul comun, după executarea funcţiei de convergenţă, să presupunem starea Z.

Atunci: D_Z = CV (X,Y)

Pentru a ilustra metoda, se consideră un exemplu, în care se dă protocolul de operare al unui DC şi se va cere întocmirea tabelului Ti, prelucrarea se în vederea obţinerii tabelelor TS şi TF.

În fig.5.2 se dă acest ptotocol de operare.

Se întocmesc dicţionarele astfel:

Starea	Nr.stării	Condiţia	Nr.condiţiei	Ieşire	Nr.ieşire
A	1	a	1	x	1
B	2	$\bar{a}$	2	y	2
C	3	b	3		
D	4	$\bar{b}$	4		
E	5	cd = 00	5		
F1	6	cd = 01	6		
F2	7	cd = 10	6		
G	8	cd = 11	7		

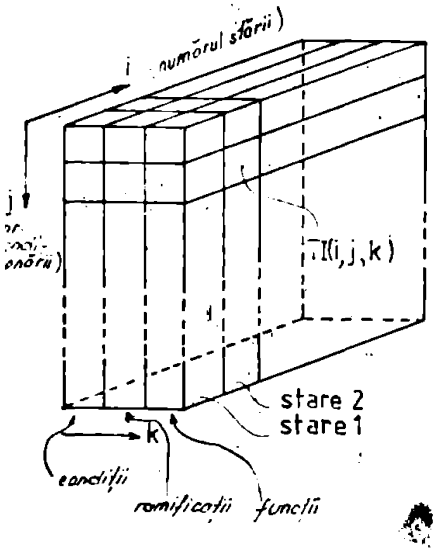
Pe baza protocolului de operare (organigrama) şi pe baza dicţionarelor se întocmeşte tabelul Ti.

120	030
210	-1
-1	020
030	-1
-1	-2
041	
-1	
352	
460	
470	
-1	
080	
-1	
510	
680	
700	
-1	

În acest tabel "-2" reprezintă sfîrşitul descrierii DC, "-1" reprezintă sfîrşitul descrierii unui pas(stare) iar fiecare linie de 3 elemente reprezintă, pentru starea curentă o <u>condiţionare</u>,<u>o stare următoare</u> şi o <u>funcţie de ieşire</u>, în această ordine

Def.formală DC

- set.intr.: $U = u_1, u_2, \ldots, u_m$
- set.stări: $X = x_1, x_2, \ldots, x_n$
- set.ieș. : $Y = y_1, y_2, \ldots, y_p$
- fcț.seov.: $X_{t+1} = \emptyset\ (U_t, X_t)$
- fcț.ieș. : $Y_t = \Psi(U_t, X_t)$

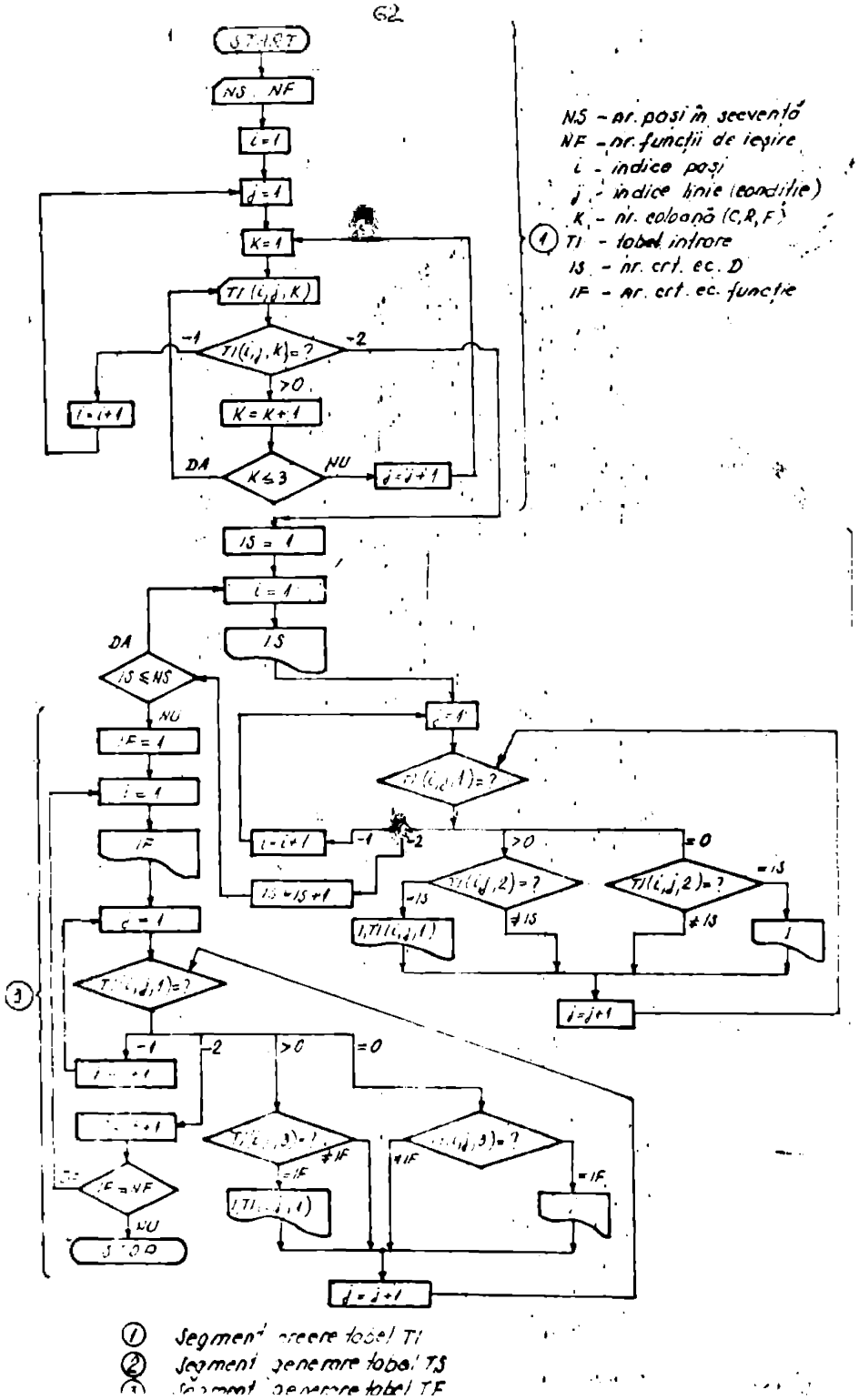


$$\begin{cases} i = 1 \div 2^8 + 1 \\ j = 1 \div 2^3 + 1 \\ k = 1 \div 3 \end{cases}$$

$$\begin{cases} TI(i,j,1)=0 \mid \Rightarrow \begin{matrix} TI(i,j,2) \\ TI(i,j,3) \end{matrix} \} \text{necond.} \\ TI(i,j,2)=0 \mid \Rightarrow \text{sfîrșit mort} \\ TI(i,j,3)=0 \mid = \text{NUL} \end{cases}$$

$$\begin{cases} TI(i,j_M+1,1)=-1 \\ TI(i_{M+1},1,1)=-2 \end{cases}$$

fig. 5.1.



① Segment creare tabel TI
② Segment generare tabel TS
③ Segment generare tabel TF

unde pe prima linie avem:

1 = Ti (1,1,1)

2 = Ti (1,1,2)

0 = Ti (1,2,3)

apoi: - 1 simbolizează sfîrşitul descrierii stării 1, urmează starea 2, etc., şi în final -2 indică sfîrşitul protocalului de operare.

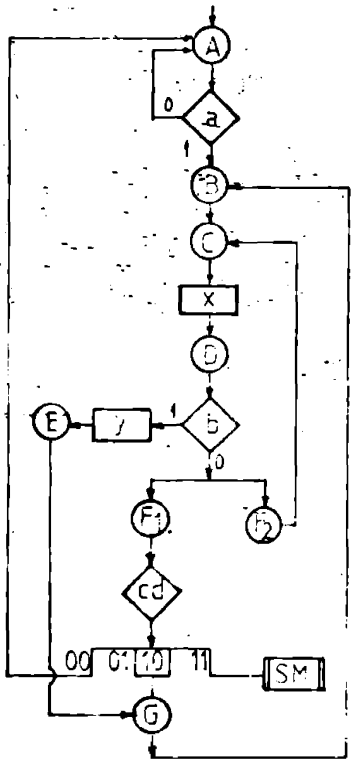


Fig. 5.2

Tabelul TS se obţine prin căutare în Ti. Astfel, se caută valoarea 1 în Ti în toate elementele sale pentru care K=2. În TS se trece valoarea 1 pentru prima intrare de date din registrul de stare urmată de perechile de puncte (i,j) pentru care K=2 şi Ti(i,j,k)=1.

Astfel se obţine TS:

1
1 2
6 5
2
1 1
3
2 0
4
3 0
5
4 3
6
4 4
7
4 4
8
5 0
6 6

Acest tabel se va transcrie astfel:

$DA = A.\overline{a} + F_1.\overline{c}.\overline{d}$

$DB = A.a$

---

unde DA este reprezentat conform dicționarului stărilor de cifra 1 din tabelul TS - prima linie, A este reprezentat de cifra 1 de pe a 2-a linie conform tot dicționarului stărilor, $\overline{a}$ este reprezentat de 2 de pe prima linie din tabelul TS conform dicționarului condițiilor, etc.

In mod similar, căutînd elementele din tabelul Ti pentru care k=3, obținem tabelul TF după cum urmează:

1
3 0
2
4 3

de unde, corelat cu dicționarul stărilor condițiilor și funcțiilor, rezultă:

$x = C$

$y = D.b$

unde:

x - este reprezentat de 1 de pe prima linie

C este reprezentat de 3 de pe a 2-a linie (0 de pe linia 2 reprezintă - necondiționare).

y - este reprezentat de 2 de pe linia 3, D - de 4 de pe linia 4 iar b - de 3 de pe linia 4.

In final, scopul a fost atins întrucît s-au obținut ecuațiile de intrare în registrul de stare și ecuațiile funcțiilor

de ieşire.

In fig.5.3 este prezentată organigrama programului de generare a tabelului Ti, de prelucrare a sa în vederea generării tabelelor TS şi TF.

Programul conţine trei segmente notate ①, ②, ③. Notaţiile utilizate au următoarele semnificaţii:

NS - număr de paşi (stări) în secvenţa protocolului de operare ;

NF - numărul funcţiilor de ieşire ;

i - indice curent pentru explorarea stărilor ;

j - indice curent pentru explorarea liniilor de condiţionare din fiecare stare ;

K - număr coloană din tabelul Ti ;

Ti - tabel intrare

iS - număr curent ecuaţie D (asigură funcţia de secvenţiere) ;

iF - număr curent ecuaţie funcţie de ieşire.

In segmentul ① al programului, se citesc NS şi NF pentru a se cunoaşte în segmentele ② respectiv ③ câte ecuaţii D va conţine TS şi câte ecuaţii de funcţii va conţine TF. Apoi se iniţializează indicii din Ti şi se citeşte primul element. Dacă elementul citit este pozitiv, se trece la citirea următorului element. Dacă elementul citit, pentru K=1, are valoarea -1 se va trece la incrementarea indicelui i - pentru o nouă stare din protocolul de operare.

In acest segment, K variază până la valoarea 3, j creşte până când Ti(i,j,1)= -1, după care este iniţializat pe 1, iar i este incrementat până când

$$Ti(i,1,1) = -2$$

In segmentul ② , se generează tabelul TS, tipărind numărul ecuaţiei D curente -iS urmat de termenii din ecuaţia sa logică daţi sub formă de perechi de puncte (i,j) pentru care:

$$Ti(i,j,2) = iS$$

Se pleacă de la IS=1 pentru prima ecuaţie D. Se consideră primul pas (stare), adică i=1 şi prima condiţionare (j=1). Se verifică primul element pentru care K=1, adică Ti(i,j,1).

Dacă:

$$Ti(i,j,1) = 0$$

nu există condiționare și deci dacă

Ti(i,j,2) = iS adică există o tranziție necondiționată spre starea cu numărul de ordine egal cu iS, termenul ce se va tipări conține numai numărul curent al stării i.

Dacă Ti(i,j,1) > 0, există condiționare, și deci dacă Ti(i,j,2) = iS, adică există o tranziție condiționată spre starea cu numărul de ordine egal cu iS, termenul care se va tipări va conține starea și condiționarea adică puncte i,j.

Dacă Ti(i,j,1) = -1, înseamnă că s-a terminat explorarea unei stări, și indicele de stări i este incrementat.

Dacă Ti(i,j,1) = -2, s-a încheiat prima explorare a Ti, pentru iS=1, și se trece la următoarea explorare, pentru iS=iS+1, etc.

Segmentul ③ al programului are aceeași structură cu segmentul ②, cu următoarele modificări :

iS este înlocuit cu iF

Ti(i,j,2) este înlocuit cu Ti(i,j,3)

în rest nu există diferențe.

In anexa 1 se află listingul programului și rezultatele obținute pentru un exemplu.

6. PROIECTAREA UNOR STRUCTURI DE DATE DUPĂ METODE SPECIFICE STRUCTURILOR DE COMANDĂ

Orice structură de date cu funcționare secvențială poate fi proiectată utilizînd metode specifice structurilor de comandă (limbaj simbolic, tabele, organigrame).

In cele ce urmează, se va exemplifica afirmația de mai sus prin două exemple:

a) Proiectarea unei scheme de convergență

In fig.6.1 este reprezentată schema bloc a unui circuit de convergență pentru două ramuri cu funcționare paralelă și modul de plasare a sa în cadrul unei organigrame de stări.

Din punctul de divergență X, urmează două ramuri paralele ce se încheie cu stările A respectiv B. Intrările în schema de convergență sînt deci A și B - ieșirile bistabililor afectați stărilor A și B în cazul automatelor cu stări necodificate, sau codurile ob inute prin produs logic între ieșirile bistabililor registrului de stare în cazul automatelor cu stări codificate. In orice caz, A și B sînt semnale cu durata

egală cu o perioadă de tact. Ieşirea schemei de convergenţă, C este tot un semnal cu durata unei perioade de tact, capabil să determine tranziţia în starea Y a automatului la începutul trunchiului comun în secvenţă. Automatul divergent şi schema de convergenţă trebuie să funcţioneze cu acelaşi tact.

De la început se poate scrie, pentru un automat cu stări necodificate:

$$D_Y = C$$

adică tranziţia în starea Y este determinată de ieşirea schemei de convergenţă.

Funcţionarea schemei de convergenţă este următoarea:

- după iniţializare, aşteaptă apariţia unuia dintre semnalele A sau B sau a ambelor semnale.

- dacă apar simultan ambele semnale, schema de convergenţă generează semnalul C, pentru că ambele ramuri cu funcţionare paralelă s-au încheiat simultan, totodată schema de convergenţă trebuie adusă în starea iniţială, în vederea asigurării unei eventuale noi funcţii de convergenţă (la reluări repetate a tranziţiei prin punctul de divergenţă).

- dacă apare numai A sau B, schema de convergenţă va trece într-o nouă stare în care aşteaptă apariţia celuilalt semnal (B sau A).

- după apariţia ultimului din cele două semnale, schema de convergenţă generează semnalul C şi se autoiniţializează.

Organigrama de funcţionare a schemei de convergenţă este prezentată în fig.6.2.

Programul în limbaj AHPL-X este prezentat în continuare

$R. \longrightarrow (\bar{A}.\bar{B}, \bar{A}B, A\bar{B}, AB)/(R,S,S,R); C = A.B$

$S. \longrightarrow (\bar{A}.\bar{B}, \bar{A}B, A\bar{B}, AB)/(S,R,R,R); C = A+B$

de unde:

$D_R = R.(\bar{A}.\bar{B}+A.B)+S(\bar{A}.B+A.\bar{B}+A.B)=R(A\odot B)+S(A+B)$

$D_S = R.(\bar{A}.B+A.\bar{B})+S(\bar{A}.\bar{B}) = R.(A\oplus B)+S\bar{A}\bar{B}$

iar ecuaţia ieşirii:

$$C = R(A.B) + S(A+B)$$

Pe baza ecuaţiilor D_R, D_S, C se poate face sinteza schemei de convergenţă, întocmai ca aceea a unui automat secven-

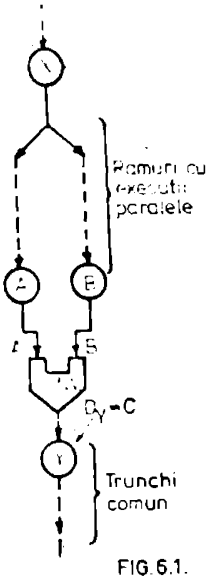


FIG.6.1.

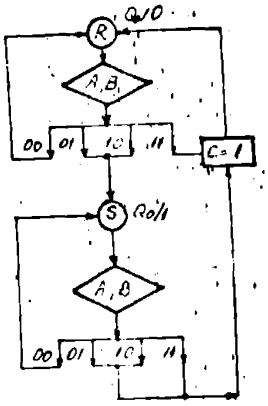


FIG.6.2

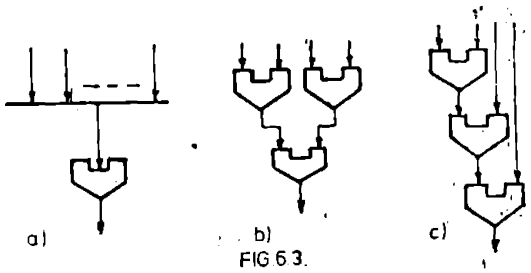


FIG 6.3.

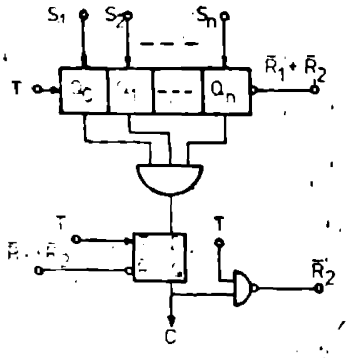


FIG. 6.4.a.

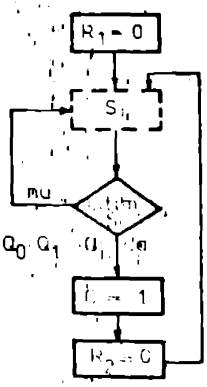


FIG 6.4 b

ţial sincron cu stări necodificate.

In cazul realizării schemei de convergenţă ca o structură de automat cu stări codificate, avînd registrul de stare dintr-un singur bistabil JK, ecuaţiile deduse pe baza organigramei de stări sînt următoarele:

$$J = A.\overline{B} + \overline{A}.B$$
$$K = A+B$$
$$C = \overline{Q}_0 A.B + Q_0(A+B)$$

unde: Q_0 este ieşirea bistabilului de stare, iar: J,K sînt intrările bistabilului de stare.

Pentru asigurarea unei convergenţe multiple (n ramuri paralele) se poate utiliza un grup de scheme de convergenţă conectate într-o structură piramidală sau serială (fig.6.3).

Structura piramidală este de preferat în cazurile în care nu există nici un indiciu asupra timpilor de parcurgere a diverselor ramuri paralele, iar structura serială se va utiliza atunci cînd există ganse să se cunoască aproximativ aceşti timpi de parcurgere. In acest caz, ramura cu cel mai lung timp presupus de parcurgere este conectată direct la ultima schemă de convergenţă. Consideraţiile de mai sus au fost făcute din motive de întîrziere de 1-2 tacte introduse de o schemă elementară de convergenţă, numărul de scheme elementare utilizate este indiferent de soluţia aleasă fiind de n-1 scheme.

O altă soluţie pentru n ramuri paralele este prezentată în fig.6.4.a, iar organigrama de principiu corespunzătoare acestei scheme în fig.6.4.b.

Fig.6.4.a conţine un registru format din n circuite basculante bistabile, fiecărei ramuri paralele fiindu-i afectat un bistabil. Iniţial bistabilele sînt poziţionate pe zero. Un semnal cu durata egală cu perioada de repetiţie a tactului, generat de ultima stare dintr-o ramură, poziţionează pe unu logic bistabilul atașat. Cînd toate aceste bistabile sînt în starea unu logic, produsul dintre ieşirile acestor bistabile are valoarea unui logic, şi la următorul impuls de tact, bistabilul C este poziţionat pe valoarea 1, indicînd momentul convergenţei complete. Ieşirea C permite intrarea în trunchiul comun ce urmează ramurilor cu execuţii paralele şi în acelaşi timp iniţializează

întreaga schemă, în vederea unei noi treceri prin ramurile cu execuţii paralele şi prin schema de convergenţă.

Organigrama din fig.6.4.b ilustrează în detaliu tocmai fenomenele ce au fost descrise mai sus.

În final, soluţia aleasă de proiectant, va depinde de viteza cerută în sesizarea convergenţei finale, şi de numărul de ramuri cu execuţii paralele.

b) Proiectarea unui circuit basculant bistabil cu funcţiuni multiple - bistabil cu tipul dictat de condiţionări externe.

Cele mai utilizate variante de circuite basculante bistabile sînt de tipurile T, D şi JK.

Vom presupune că dorim să realizăm un bistabil condiţionat care după dorinţă, să se comporte fie ca bistabil T, fie D fie JK. În literatură nu există o metodă unitară de proiectare a unor astfel de bistabile condiţionate.

În cele ce urmează, se va proiecta un astfel de bistabil considerîndu-l ca un caz particular de dispozitiv de comandă de tip sincron, fără funcţii de ieşire şi fără registru de stare. Rolul registrului de stare fiind preluat de însăşi bistabilul considerat, la bază fiind iniţial, spre exemplu de tipul D.

Schema bloc a unui astfel de bistabil condiţionat este prezentată în fig.6.5, ce şi diagramă de timp ce trebuie respectată, referitor la impulsul de sincronizare, intrările de condiţii şi intrările de date. În fig.6.5, notaţiile au următoarea semnificaţie:

Q_o, $\overline{Q_o}$ - ieşirile bistabilului ;

AZ - intrare asincronă de iniţializare ;

BT - intrare de condiţie ce impune ca bistabilul să aibă o funcţionare corespunzătoare tipului T ;

BD - intrare de condiţie corespunzătoare tipului D ;

BJK- intrare de condiţie corespunzătoare tipului JK;

ID - intrare de date sincrone pentru tipurile T şi D;

I1,I0 - intrare de date sincrone pentru tipul JK (eventual RS); I1 - determină trecerea în starea 1 a bistabilului, iar I0 - determină trecerea în starea 0.

p - impulsul de sincronizare (tact).

In cadrul cronogramei:

X = T,D,JK (ex.dacă X=T, atunci BX=BT)
Y = D,1,0 (ex.dacă Y=D, atunci IY=ID)

La un moment dat, bistabilul poate funcţiona conform unui singur tip din cele considerate (T,D sau JK). Această afirmaţie se exprimă prin expresia:

$$BT.BD + BT.BJK + BD.BJK = 0$$

Organigrama de stări a unui astfel de bistabil este prezentată în fig.6.6, unde cu A şi B sînt notate cele două stări posibile ale bistabilului (0 sau 1). Pentru lămurirea funcţionării bistabilului pe baza organigramei vom urmări o singură ramură(cea dublată cu linie întreruptă). Dacă bistabilul este în starea 1 (B) şi BT=0 şi BD=0 şi BJK=1 înseamnă că se cere ca acest bistabil să funcţioneze conform tipului JK. Dacă intrarea de zero are valoarea 1 logic (IO=1), bistabilul va comuta în starea zero (A).

Conform procedurilor standard de proiectare, în fig. 6.7 sînt prezentate diagrama stărilor curente şi diagrama stărilor următoare.

Pe baza diagramei stărilor următoare, rezultă ecuaţia:

$$D_Q = \overline{Q_0}(BT.ID + \overline{BT}(BD.ID + \overline{BD}(BJK.I1))) +$$
$$+ Q_0(BT.\overline{ID} + BT(BD.ID + \overline{BD}(BJK.\overline{IO} + \overline{BJK})))$$

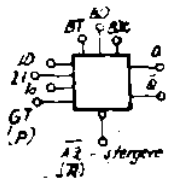
prin descompunere rezultă forma:

$$D_Q = \overline{Q_0}.BT.ID + \overline{Q_0}.\overline{BT}.BD.ID + \overline{Q_0}.\overline{BT}.\overline{BD}.BJK.I1 +$$
$$+ Q_0.BT.\overline{ID} + Q_0.\overline{BT}.BD.ID + Q_0.\overline{BT}.\overline{BD}.BJK.\overline{IO} +$$
$$+ Q_0.\overline{BT}.\overline{BD}.\overline{BJK}$$

Minimizarea acestei ecuaţii *punînd* probleme datorită numărului mare de variabile, se va face o minimizare pentru termenii ce conţin variabila I1 şi una pentru termenii ce conţin IO. În final se vor verifica ecuaţiile obţinute.

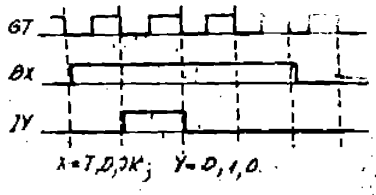
Cele două diagrame de minimizare sînt prezentate în fig.6.8. Termenii redondanţi rezultă din sumele de produse logice ale condiţionărilor simultane interzise.

În urma minimizării şi unificării ecuaţiilor rezultă expresia finală:



ID - intrare de date T sau D
I1 - intrare sincronă de punere pe "1" (J).
ID - intrare sincronă de punere pe "0" (K)

Fig.6.5



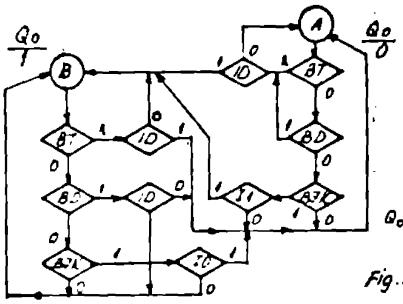


Fig.6.6

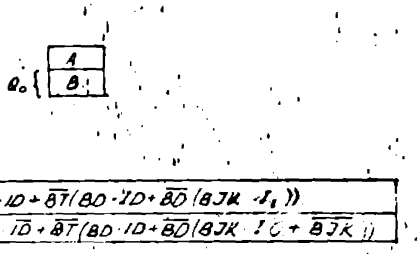


Fig.6.7.

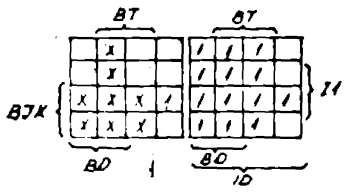
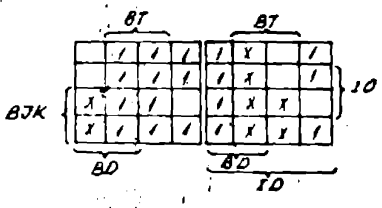


Fig.6.8.



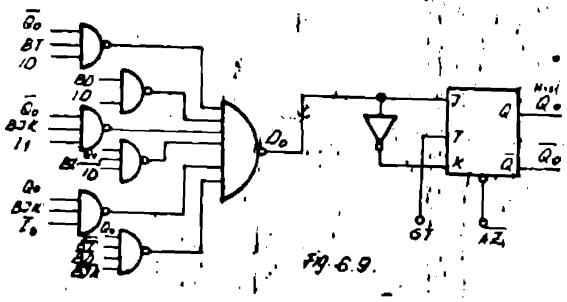


Fig.6.9.

$$D_o = \overline{Q_o}.BT.1D + \overline{Q_o}.BD.ID + \overline{Q_o}.BJK.I1 +$$
$$+ Q_o.BT.\overline{ID} + Q_o.BD.ID + Q_o.BJK.\overline{IO} + Q_o.\overline{BT}.\overline{BD}.\overline{BJK}$$

Grupînd termenii 1 şi 4 respectiv 2 şi 5 rezultă:

$$D_o = BT(Q_o \oplus ID) + BD.ID + \overline{Q_o}.BJK.I1 + Q_o.BJK.\overline{IO} +$$
$$+ Q_o.\overline{BT}.\overline{BD}.\overline{BJK}$$

Pe baza acestei ecuaţii s-a făcut sinteza schemei bistabilului condiţionat, prezentată în fig.6.9. Schema este experimentată, funcţionînd perfect.

Similar, pot fi proiectate scheme care funcţionează în orice mod posibil, ca de exemplu: un bistabil care iniţial - la primul impuls de tact are o funcţionare de tip D, la următorul impuls de tact, funcţionează ca un bistabil T, apoi ca JK, etc.

7. SEGMENTAREA ŞI CUPLAREA DISPOZITIVELOR DE COMANDA

In literatura de specialitate sînt puţine referiri la funcţionarea în serie sau în paralel a mai multor dispozitive de comandă [C18, H7] . Funcţionarea paralelă este similară funcţionării în cazul existenţei mai multor ramuri divergente într-un DC. Problema divergenţei şi a convergenţei mai multor ramuri paralele a fost tratată în capitolele anterioare.

Funcţionarea serială a mai multor DC este tratată în [C18] cînd cuplarea DC este asigurată prin intermediul intrărilor şi ieşirilor (generare funcţie într-un DC şi verificarea acestei funcţii ca intrare în alt DC). Metoda este perfect valabilă cînd mai multe DC care trebuie să coopereze prin funcţionare serială, utilizează aceleaşi impulsuri de tact. Dacă se utilizează impulsuri de tact diferite ca frecvenţă de repetiţie, atunci DC cu tactul mai rapid riscă să nu i se ia în considerare o funcţie de ieşire pentru cuplare cu alt DC. In acest caz organigrama de funcţionare a DC cu tact de mare frecvenţă se complică, în sensul că după lansarea comenzii de cuplare, trebuie să verifice dacă a fost luată în considerare, dacă nu, trebuie să repete comanda pînă cînd aceasta este în sfîrşit luată în considerare.(In cazul acesta se poate pierde foarte mult timp). Cînd auto-

matul mai lent execută comanda de cuplare, nu se pun astfel de probleme, caz similar cu cazul cînd ambele DC au acelaşi tact. In orice caz, în situaţia DC rapide, care cer cuplarea cu automate mai lente, se complică însăşi organigrama de funcţionare a primei categorii.

Dacă două automate ce trebuiesc cuplate, au frecvenţele de repetiţie a impulsurilor de tact necunoscute, atunci ambele organigrame de funcţionare se vor complica în mod inutil

In cazul în care există mai multe automate organizate într-o anumită ierarhie, pe două sau mai multe nivele, cu funcţionare întreţesută serie/paralel, metoda de cuplare la nivelul lansare comandă - verificare întrare, este practic insabordabilă, datorită complicării fiecărui protocol de operare, în special în cazul în care relaţiile de frecvenţă de repetiţie între diferite impulsuri de tact sînt necunoscute sau variabile.

In literatura analizată [C18, H7, C19, I1, K4] nu se prezintă probleme de segmentare a automatelor sau dispozitivelor de comandă.

In cele ce urmează, vom prezenta <u>o metodă de segmentare şi cuplare a automatelor</u>, ce nu modifică şi nu complică în mod inutil protocoalele de operare ale acestora (organigramele sau tabelele), cu funcţionare sigură, ce nu pune probleme de hazard, indiferent de tipul de ierarhie utilizat la cuplare, funcţionare paralelă, serie sau întreţesută şi de asemenea independentă de frecvenţele impulsurilor de tact ale diverselor automate sau segmente de automate.

In cazul în care, pentru rezolvarea unei probleme complexe de comandă numerică, apare necesitatea proiectării şi realizării unui ASS de mare anvergură (foarte multe stări, bucle, ramificaţii, funcţii), se preferă a împărţi sau segmenta automatul în mai multe segmente sau automate relativ simple. Tehnica este întrucîtva similară tehnicii segmentării programelor, respectînd principii asemănătoare.

- segmentele să fie ramuri distincte ale ASS fără salturi în alte ramuri ;
- segmentarea se va face în afara buclelor de aşteptare sau ciclare, şi nu în interiorul unei bucle.
- se permite segmentarea în interiorul unei bucle numai

dacă punctul terminal al segmentului se va afla în cadrul aceleiaşi bucle.

- în plus, faţă de tehnica segmentării programelor, putem avea mai multe segmente cu funcţionare simultană total sau parţial, dacă nu se ajunge la un conflict referitor la resursele hardware afectate.

In cap.6 vor fi analizate diverse moduri de ierarhizare în cadrul mai multor ASS segmentate şi cuplate, verificate experimental. De asemenea vor apare cazuri de funcţionare serială, paralelă, întreţesută, cu unul sau mai multe automate MASTER, precum şi interschimbabilitatea MASTER-SLAVE, multiplexată în timp. De asemenea vor fi prezentate înlănţuiri de tipul: apel-apel-apel....-revenire-revenire-revenire, întocmai ca la buclele DO cuibărite, cunoscute spre exemplu din limbajul FORTRAN.

Problema cuplării ASS apare nu numai în cazul segmentării ci şi în cazul când avem deja realizate mai multe DC care trebuie să funcţioneze corelate.

Putem avea următoarele situaţii:

- un automat principal şi mai multe automate subordonate ;

- mai multe automate din care la un moment dat, unul este principal şi celelalte sînt subordonate, la un alt moment, un alt automat devine principal şi restul subordonate, etc.

- mai multe automate principale şi mai multe automate subordonate, cu execuţii simultane sau întreţesute.

Mai jos, nu vom analiza decît tehnica cuplării a două unităţi de comandă notate UC1 şi UC2, sub formă de ASS, tehnică propusă şi experimentată de autor.

Cuplarea este asigurată de către un bistabil de legătură (sau de cuplare) BL, şi decurge conform fig.7.1.

La aplicarea unui semnal de stare iniţială (SIN) bistabilul BL este adus la 0, UC1 trece în starea iniţială K, iar UC2 în starea iniţială X. UC1 parcurge stările K...A realizîndu-şi o parte din protocolul de operare (etapa I1). După starea A, UC1 trebuie să-şi oprească funcţionarea predînd comanda lui UC2. Acest lucru se întîmplă astfel: în starea A, UC1 comandă poziţionarea bistabilului, BL pe 1, prin semnalul BL←1. Pînă în acest moment, UC2 este în buclă de aşteptare

Fig. 7.1.

FIG 7.2.

în starea X, pentru BL=0. Din moment ce BL=1, UC2 va începe să își realizeze sarcine, preluînd comenzi și parcurgînd stările Y...Z (etapa II) Între timp, UC1 rămîne în buclă de așteptare în starea B pentru că acum BL=1. După ce UC2 și-a încheiat execuțarea protocolului său de operare, va trebui să realizeze încă două operații specifice numai cuplării:

- autoinițializarea UC2 și blocarea ei în starea inițială X ;
- ștergerea bistabilului de cuplare, în vederea redeclanșării UC1.

Cînd BL=0, UC1 părăsește bucla de așteptare din starea B, executînd tranziția în starea C, după care își continuă execuția protocolului său de operare (etapa I_2).

Dacă în starea A, UC1 efectuează și alte operații din cadrul protocolului său de operare (în afara generării comenzii BL←1), etapa este renotată cu I_1'.

Dacă în starea Z, UC2 efectuează și alte operații din cadrul protocolului său de operare (în afara generării comenzii BL←0), etapa este renotată cu II'.

Alegerea variantelor I_1' și II' în locul variantelor I_1 și II conduce la eliminarea unor perioade de 1 tact inactive, în funcționarea corelată a celor două automate, după cum se va vedea în fig.7.2.

În cazul considerat, din cele două automate, UC1 este principal, iar UC2 subordonat.

Chiar dacă se iau în considerare mai multe automate sau segmente de automate, tehnica de cuplare prezentată mai sus poate fi utilizată fără probleme, indiferent de raporturile dintre perioadele impulsurilor de tact și ierarhia stabilită între automate.

De exemplu, este posibilă o funcționare întrețesută în timp a mai multor automate, ca în fig.7.3, aleasă arbitrar pentru a ilustra posibilitățile tehnicii adoptate. Cazuri concrete vor fi analizate în cap.8.

UC1 este un automat principal segmentat în segmentele 1.1 - 1.2 și 1.3 - 1.4 (prevăzut cu sfîrșit mort)

UC2 este un alt automat principal dintr-un singur segment cu autorevenire la starea inițială (2.1 - 2.2).

UC3 este un automat subordonat lui UC1, care pe durata

intervalului 3.1 - 3.2 este în buclă de aşteptare a comenzii din punctul 1.2 al UC1. Perioada sa activă este cuprinsă între punctele 3.2 - 3.3, cu revenire în starea iniţială, după ce în punctul 3.3 predă comanda lui UC1 (segmentul 1.3 - 1.4).

UC4 este din nou un automat principal cu ramuri paralele unde:

4.1 - este bucla de aşteptare iniţială
4.1 - 4.2 - trunchi comun
4.2 - 4.3 - sfîrşit mort (UC_4^1)
4.2 - 4.4 (UC_4^1) ramuri paralele
4.2 - 4.4 (UC_4^2)
4.4 - 4.5 - trunchi de aşteptare a convergenţei
4.5 - 4.6 - ultimul trunchi comun

UC5 este un automat subordonat lui UC4, care pe durata intervalului 5.1 - 5.2 este în buclă de aşteptare a unei comenzi de declanşare din partea lui UC1. Această comandă se declanşează în punctul 4.5. De remarcat faptul că între aceste două automate nu există un dialog în adevăratul sens al cuvîntului, pentru că UC4 şi UC5, din acest moment vor funcţiona în paralel, fără ca UC4 să mai controleze secvenţierea lui UC5 după ce l-a declanşat. In acest caz, bistabilul corespunzător de cuplare, va fi şters la iniţializarea generală, şi va fi trecut în starea 1, de către UC4 în punctul 4.5 (după asigurarea convergenţei).

S-a insistat pînă acum asupra cuplării DC, a funcţionării lor în paralel, şi datorită faptului că previziunile pentru viitor [54], referitoare la generaţia a 5-a de calculatoare întrevăd atît la nivel software - limbaje de programare orientate spre obiect şi cu funcţionare în paralel, cît şi la nivel hardware - arhitecturi de procesoare în paralel.

Şi din acest motiv, studiul comenzii paralele este util atît pentru etapa actuală de dezvoltare a tehnicii numerice cît şi pentru etapele viitoare.

O aplicaţie posibilă şi imediată pentru utilizarea unui sistem de DC cuplate şi cu funcţionare în paralel ar putea fi de exemplu la realizarea DC central de comandă pentru un procesor cu aducerea anticipată a instrucţiunilor şi datelor şi cu un număr mare de tipuri de cicluri maşină. In acest caz, DC central de comandă ar putea avea o structură de tipul celei

prezentate în fig.7.4.

Aici se disting 4 DC cuplate, cu funcţionare în paralel din care:

- DC1 - serveşte la generarea microinstrucţiunilor sau semnalelor de comandă pentru executarea instrucţiunilor ;
- DC2 - constituie generatorul noului ciclu de maşină (eventual şi "codificator de ciclu maşină") - sau generator de "faze de execuţie".
- DC3 - serveşte la generarea comenzilor necesare pentru aducerea anticipată a instrucţiunilor.
- DC4 serveşte la aducerea anticipată a datelor (ex.:elemente succesive dintr-un tablou),

De remarcat în fig.7.4 modul cum sînt cuplate între ele cele 4 DC în aşa fel încît se condiţionează unul pe celălalt. Semnalul de tact poate fi unic sau tactul pentru DC2 şi DC3 poate fi derivat din tactul pentru DC1 (la DC2 divizat) sau autorizat prin poartă (la DC3).

În cazuri mai complexe de procesări sau execuţii paralele, poate exista conflict de cuplare la nivelul mai multor DC sau la nivelul altor resurse hardware implicate în cuplare. Pentru rezolvarea unor astfel de probleme, propunem o structură (fig.7.5), ce conţine următoarele blocuri în secţiunea sa de date:

RCC - registrul cererilor de cuplare a DC-urilor ;
RDA - registrul DC-urilor active ;
RDB - registrul DC-urilor inactive (în aşteptare)
RBC - registrul bistabililor de cuplare
SPC - schemă de priorităţi la cuplare
SCC - schemă de convergenţă ; asigură cuplarea unui DC care aşteaptă încheierea unui grup de execuţii simultane în alte DC. Pentru mai multe grupe de execuţii simultane sînt necesare mai multe SCC.
DCC - dispozitiv de comandă a proceselor de cuplare, este de fapt un DC supervizor al celorlalte DC şi a blocurilor funcţionale din structura propusă în fig.7.5.

Evident, proiectarea protocolului de operare al DCC nu este o treabă simplă, pentru că ea depinde de numărul de DC controlate, de ierarhizarea lor, de celelalte resurse hardware implicate, de numărul de grupe de execuţii paralele, etc.

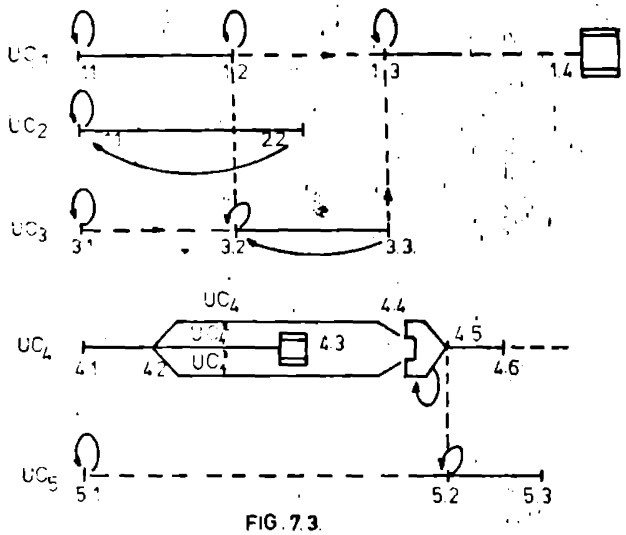


FIG. 7.3.

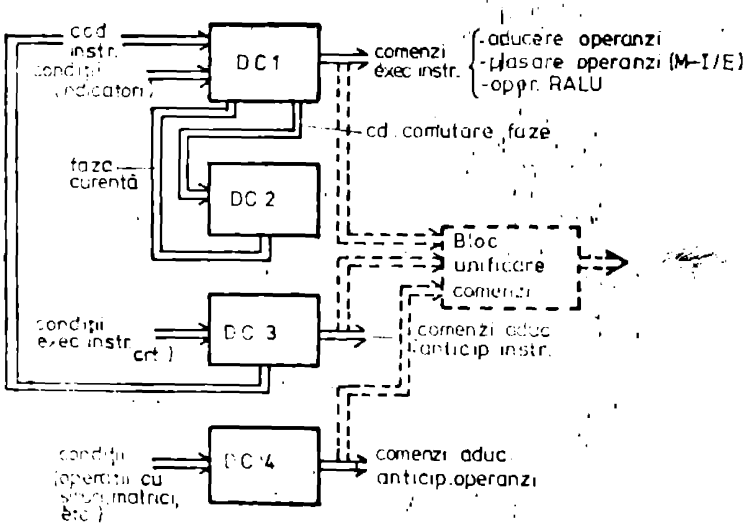


FIG. 7.4.

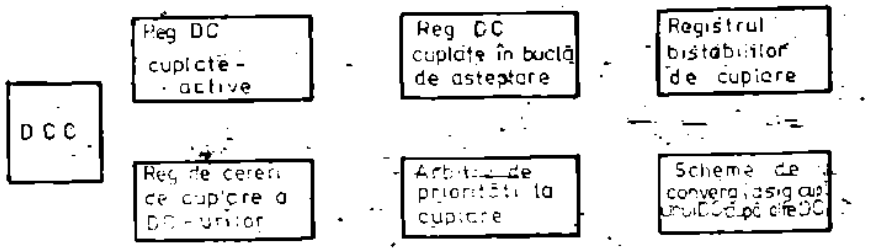


FIG. 7.5.

Astfel, de probleme, preocupă şi pe cei ce propun structuri de SN din generaţia a 5-a [*4] , şi posibil ca în viitor să se elaboreze proceduri standard de proiectare a unor astfel de structuri, aşa cum astăzi există proceduri standard de proiectare a numărătoarelor, schemelor combinaţionale, automatelor, sau a calculatoarelor în arhitectură Von Neumann.

8. REALIZĂRI HARDWARE

In acest capitol se vor prezenta unele realizări hardware cu caracter de originalitate, în care se regăsesc, de astădată demonstrate practic, principiile enunţate în cap.4 şi 7. Blocurile de comandă ce urmează a fi prezentate, au fost realizate de autor în cadrul unor contracte de cercetare, pentru structuri de date deosebit de complexe. Soluţiile alese reprezintă soluţii optime de compromis, în condiţiile şi la cerinţele impuse (viteză de operare, caracteristici de "timp real", execuţii paralele, etc.).

8.1. Structură de comandă ierarhizată supervizată de un sistem monoprocesor, utilizată în cadrul unui stand de testare a memoriilor.

Pentru a analiza structura de comandă din titlu se cere a fi prezentată succint şi structura de date aferentă şi funcţiunile sale. Standul de testare este condus de un microprocesor şi poate testa atît memorii cu ferite cît şi memorii cu circuite MOS, pînă la 4 module simultan, de fiecare tip (ferite, MOS), cu capacităţi de 8, 16, 32 şi 64 KØ (Anexa 2, FOTO 1).

La o analiză globală, standul de testare conţine trei secţiuni (fig.8.1.a).

1 - un sistem de calcul cu periferia (DAF, panou de comandă şi vizualizare - PCV, panou de service - PS , miniimpri-

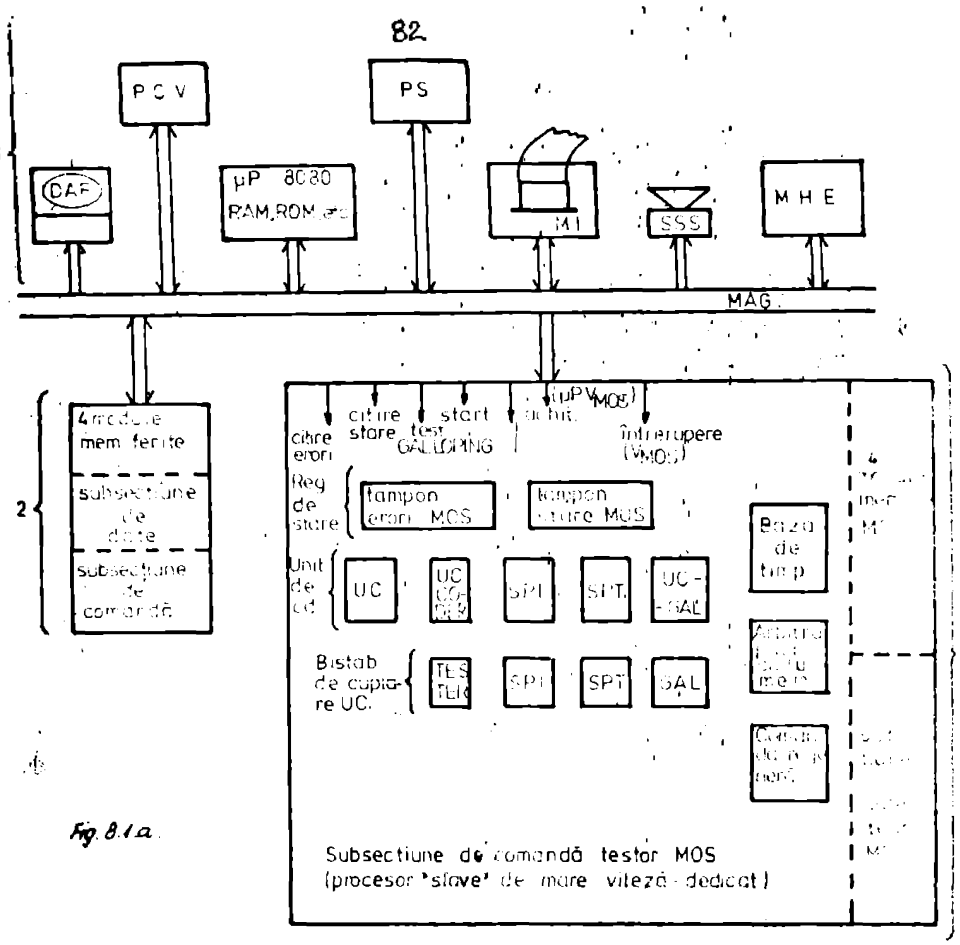


Fig. 8.1.a.

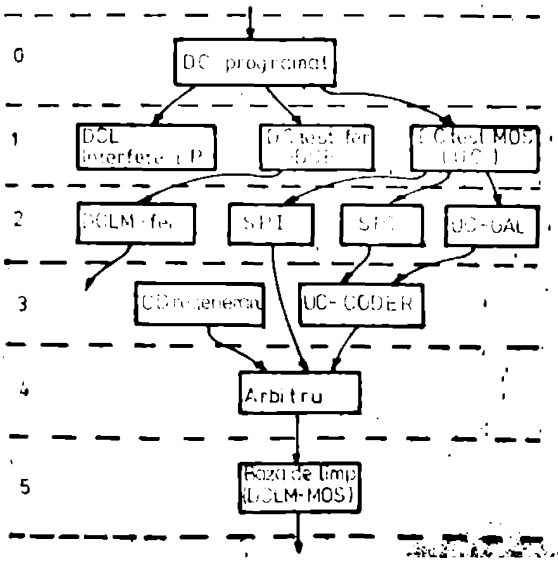


Fig. 8.1.b.

mantă - MI, sintetizor semnale sonore - SSS, monitor hardware pentru electroalimentare - MHE).

2 - testorul pentru memoriile cu ferite ;

3 - testorul pentru memoriile MOS .

Secţiunea 1 este secţiune "master" avînd întreaga comandă de tip "programat".

Secţiunea 2 este o secţiune "slave" şi conţine 4 module de memorie cu ferite, o subsecţiune de date aferentă structurii de testor şi o subsecţiune de comandă de tipul prezentat în cap. 3.1.2.2.

Secţiunea 3 este o altă secţiune "slave" şi conţine 4 module de memorie MOS, o subsecţiune de date aferentă structurii de testor şi o subsecţiune de comandă, ierarhizată pe mai multe nivele, detaliată şi comentată în *fig.3.1.a.*

In fig.3.1.b este prezentată ierarhizarea standului la nivelul comenzii, unde pe nivelul zero (master) se află dispozitivul de comandă programat, pe nivelul 1 dispozitivele de comandă centrale ale celor două testoare: DCF - pentru testorul de memorii cu ferite, UC - pentru testorul de memorii MOS plus DCL hardware interfeţe. Următoarele nivele ierarhice se continuă după cum urmează:

- nivelul 2 conţine : SPI (unitatea de comandă pentru regimul de îmbătrînire - Subprogram îmbătrînire); SPT (unitatea de comandă pentru testele curente - Subprogram testare); UC-GAL (unitatea de comandă ce dirijează testul de durată mare - GALLOPING) şi DCLM - ferite.

- nivelul 3 conţine : UC-GOBER (unitatea de comandă ce dirijează operaţiile de citire/comparare memorare cod eroare) şi comande regenerării.

- nivelul 4 conţine: un arbitru de concurenţă/priorităţi între cererile de cicluri citire/scriere/regenerare).

- nivelul 5 conţine: BT (baza de timp pentru modulele de memorie MOS) un DC de tip asincron realizat cu linii de întîrziere, porţi logice şi bistabile.

Dialogul între blocurile de mai sus este de tipul:"cu respectarea ierarhiei".

Pentru a efectua analiza şi sinteza nivelelor ierarhice 1 - 5 se impune prezentarea globală a protocolului de operare şi a secţiunilor de date implicate.

Testorul MOS execută sub comanda DC progr·mat (nivel ierarhic 0) următoarele funcţiuni majore:

- cicluri de îmbătrînire
- cicluri de testare curente: scriere-citire "0"; scriere-citire "1"; scriere/citire informaţia de adresă; scriere/citire informaţie poziţionată manual (de la un registru de comutatoare).
- cicluri de testare în regim "GALLOPING".

Testorul are următoarele facilităţi:

- oprire pe eroare
- avans manual în secvenţă (pas - cu - pas)
- afişare adresă, date de I/E, rezultatul comparării,etc.

In continuare, pe baza unei scheme bloc referitoare numai la testorul MOS, se va explica principiul de funcţionare, se vor descrie blocurile din secţiunea de date şi se vor descrie şi proiecta unităţile de comandă.

Schema bloc propusă este cea din figura 3.1.c. Modulul de memorie din figură, prin intermediul multiplexorului de intrare adresă MUX - IA poate fi adresat de către trei blocuri diferite: numărătorul N_2 (numărătorul de adresare curentă); numărătorul N_1 (utilizat doar în cadrul testului GALLOPING),sau numărătorul de regenerare N-REG (formează adresa în timpul ciclului de regenerare).

Semnalele prin care multiplexorul MUX - IA permite ca adresarea să se efectueze de către unul dintre numărătoare sînt generate (ca nivele logice) de către un generator de faze pentru multiplexoare GF - IA.

Datele ce urmează a fi înscrise în modul sînt şi ele multiplexate din trei surse diferite:

- conţinutul numărătorului N_2 pentru memorarea adresei curente.
- conţinutul cheilor de date de pe panoul de comandă.
- conţinutul bistabilului D ("0" sau "1").

Ca şi multiplexorul de intrare adresă MUX - IA şi multiplexorul de intrare date MUX - ID este comandat de un generator de faze pentru multiplexare: GF - ID.

Sesizarea unei erori de memorare se face prin compararea datelor de memorat cu cele memorate, cu ajutorul comparatorului COMP. Rezultatul pozitiv al comparării va fi semnalat de apari-

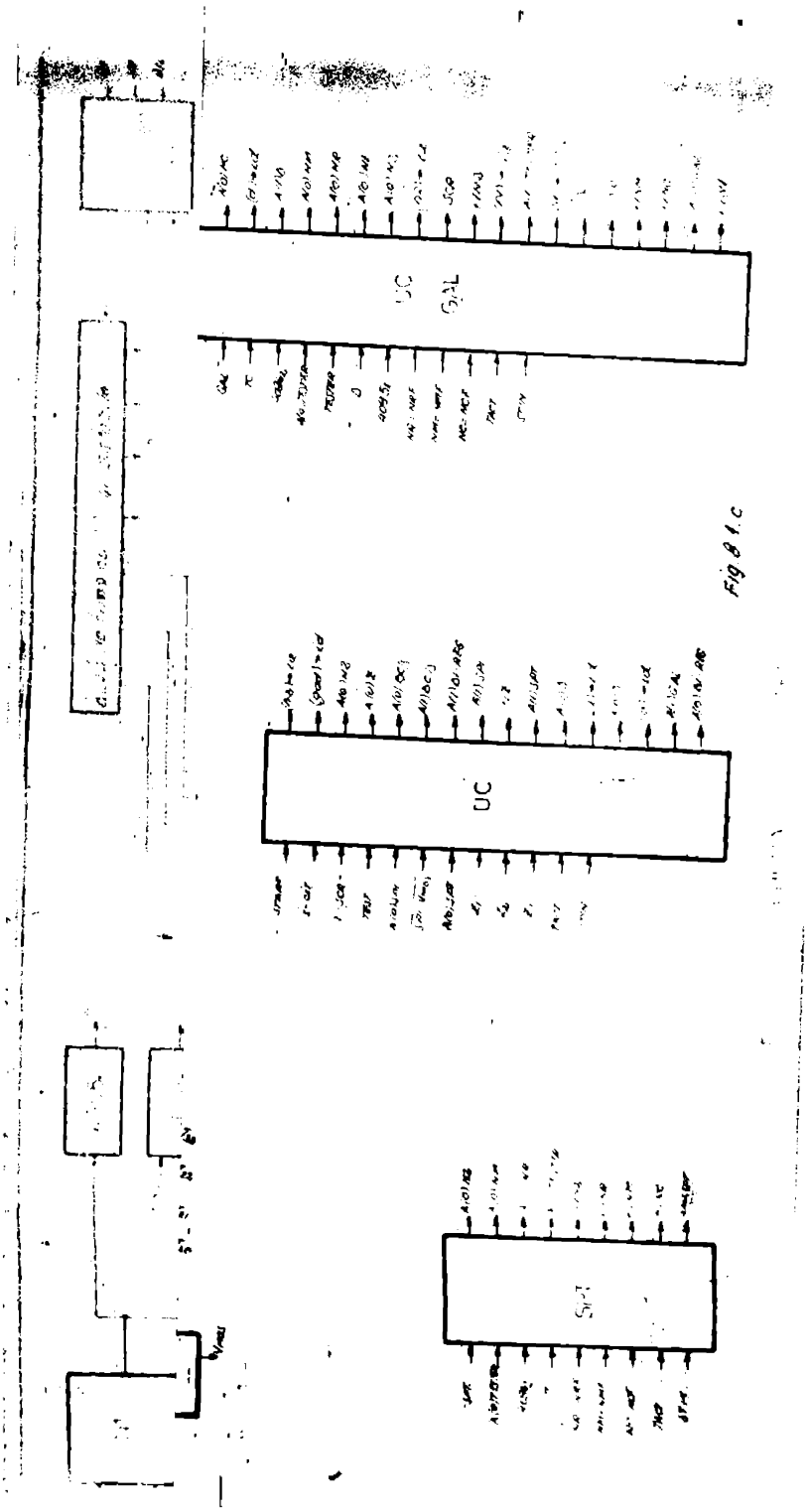


Fig 8.1.c

:omba-

:orul
:F şi
:a:

:scria

:a mo-
:a che-
:ei, se
: număr
COD -
- NR
:rge-
:az de
de
pe
:gal

:ta,
:e
:e pot
:e co-
l
: în
: NC,
lu.

:erea
:zen-
l.
:ului
:OMP-
:nţă),
:umă-

: din-

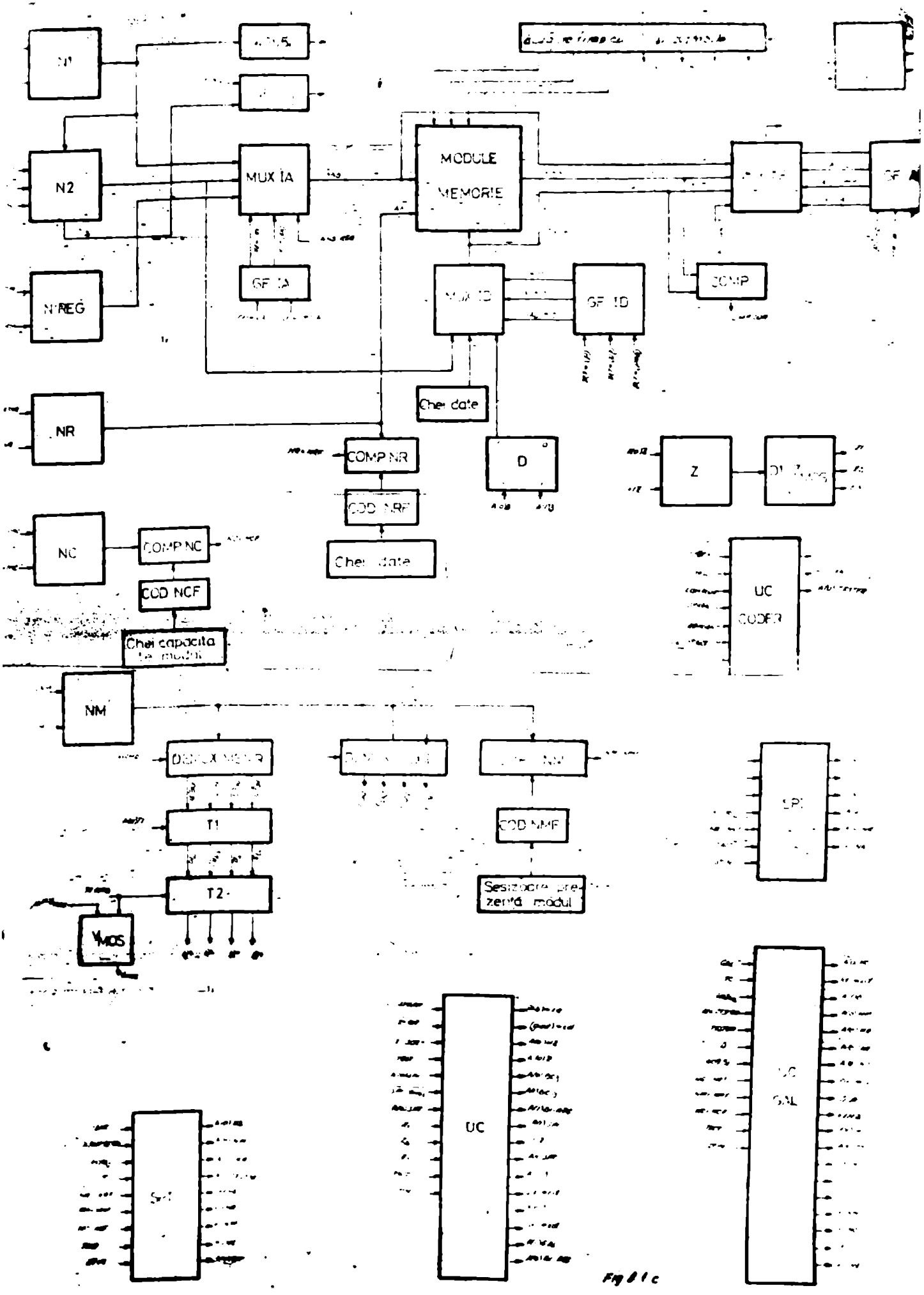

N2
MUX IA
MODULE
MEMORIE
N'REG
GF IA
MUX ID
GF ID
COMP
NR
Chei date
COMP NR
D
Z
NC
COMP NC
COD NCF
Chei capacita
UC
CODER
NM
T1
T2
YMOS
COD NMF
SPI
UC
UC
GAL

Fig 8 1 c

ţia semnalului COMP COR (comparare corectă). Rezultatul comparării bit cu bit este de asemenea accesibil.

Sistemul de vizualizare (materializat prin multiplexorul de afişare MUX - AF, generatorul de faze asociat: GF - AF şi un set de LED-uri pe panoul frontal), permite vizualizarea:

- datelor citite
- adresei curente
- rezultatului comparării bit cu bit a datelor de înscris şi cele înscrise.

Capacitatea modulelor poate fi variabilă, capacitatea modulului aflat în testor fiind consemnată prin poziţionarea cheilor cu această destinaţie. De fapt poziţionînd aceste chei,se reţine numărul rîndurilor de capsule ale modulului. Acest număr este codificat (de către codificatorul de rînduri final: COD - NRF) apoi comparat prin intermediul comparatorului COMP - NR cu conţinutul numărătorului NR (NR numără în timpul parcurgerii unui test, rîndul de capsule la care s-a ajuns). In caz de coincidenţă între conţinutul numărătorului NR şi numărul de rînduri (capacitatea) fixată prin intermediul cheilor de pe panou, apare semnalul NR = NRF (numărul de rînduri este egal cu numărul de rîndul final).

Testorul are posibilitatea de a cicla diferitele teste, numărul de cicluri care trebuie parcurs specificîndu-se de către operator prin intermediul unor chei de la panou. (Se pot executa 1,2,4 sau 8 cicluri). Conţinutul acestor chei este codificat de către codificatorul numărului de cicluri final (NCF), apoi introdus în comparatorul numărului de cicluri în vederea comparării cu conţinutul numărătorului de cicluri NC, numărător incrementat după fiecare parcurgere a unui ciclu. Coincidenţa este semnalată de către semnalul NC = NCF.

Numărul de module ce sînt introduse în testor în vederea testării lor, este trimis (de către patru sesizoare a prezenţei modulului), codificatorului numărului de module final. Conţinutul acestuia este comparat cu conţinutul numărătorului de module NM, de către comparatorul numărului de module COMP-NM. Dacă s-a ajuns la ultimul modul (deci există coincidenţă), apare semnalul NM - NMF (numărul de module este egal cu numărul de module final).

In cazul apariţiei unei erori, deci a necoincidenţei din-

tre datele de intrare şi cele memorate, acest lucru trebuie semnalat sistemului de conducere (microprocesorului). Trebuie totodată indicat modulul la care a apărut defecţiunea respectivă. Acestea toate se realizează astfel: tamponul T_1, conţine patru circuite bistabile, cîte unul pentru fiecare modul. Conţinutul numărătorului de module NM serveşte la demultiplexarea de către demultiplexorul memorare eroare DEMUX - MEMER a semnalului MEMER. La apariţia lui, bistabilul asociat modulului (bistabil din tamponul T_1) la care a intervenit eroarea, se va poziţiona pe "1".

Conţinutul tamponului T_1 este transferat tamponului T_2, a cărui ieşiri sînt conectate la bus -ul de date al microprocesorului, pentru a se putea trimite informaţia asupra modulului defect. Transferul se face la apariţia semnalului T_1 în T_2 (semnal identic cu cel lansat de către microprocesor: citire eroare MOS - CITER MOS).

După executarea transferului, microprocesorul trebuie anunţat că i s-a furnizat informaţie pe bus-ul de date. Aceasta se face lansîndu-se o întrerupere, prin intermediul bistabilului V_{MOS} (validare MOS). Bistabilul V_{MOS} este poziţionat pe "1" în momentul transferului T_1 în T_2. Semnalul de achitare a întreruperii $\mu P - V_{MOS}$ este cel care readuce pe "0" acest bistabil, permiţîndu-se continuarea testării.

Demultiplexorul DEMUX - NDOI, demultiplexează semnalul $NDOI_{1,2}$ spre cele patru module, folosindu-se de numărătorul modulului curent NM.

Pentru parcurgerea operaţiunilor de testare descrise mai sus e nevoie de un bloc ce să evidenţieze execuţia curentă a unuia sau altuia dintre teste. Acest bloc este un numărător, numărătorul de teste Z. Conţinutul acestuia este:

0 0 0 la îmbătrînire
0 0 1 la testul scriere, citire, comparare cu conţinutul cheilor de date
0 1 0 la testul scriere, citire, comparare cu "0"
0 1 1 la testul scriere, citire, comparare cu "1"
1 0 0 la testul scriere, citire, comparare adresă curentă
1 0 1 la testul GALLOPING

In schema din figura 8.1.e sînt figurate şi cinci automate secvenţiale sincrone, materializarea celor cinci unităţi de

comandă ale testorului, unităţi ce generează diversele semnale de comandă pe parcursul testelor.

UC - unitatea de comandă centrală; ce asigură iniţializarea întregului testor, declanşarea regimului de îmbătrînire alea de operator (citire/scriere/regenerare), declanşarea succesivă a testelor.
Tot UC asigură avansul numărătorului de teste Z, poziţionarea bistabilelor BC/S (citire/scriere), D(date "1" sau "0") şi fixarea intrărilor de selecţie a multiplexorului de intrare date MUX-ID.

UC-GAL - unitatea de comandă pentru testul GALLOPING. Determină iniţializarea numărătoarelor de cicluri NC, de zodule NM, de rînduri NR, a numărătoarelor de adresare N_1 şi N_2 ; determină selectarea adresei prin multiplexorul de intrare adresă MUX-IA; incrementează numărătoarele NR, NM, NC, N_2; determină transferul conţinutului numărătorului N_1 în numărătorul N_2; dă comanda de poziţionare pe "1" a bistabilului de date D, precum şi comanda de complementare a datelor, deci generează toate semnalele necesare desfăşurării operaţiunilor testului GALLOPING.

UC-CODER - Unitatea de comandă ce asigură formarea codurilor de eroare în tamponul T_1. Declanşează un ciclu de citire-comparare şi după trecerea timpului de acces al modulelor, în funcţie de rezultatul comparării memorează sau nu codul de eroare. Semnalele de comandă efective pe care le generează, sînt: de iniţializare a bistabilului BESTAB, de citire şi de memorare a erorii.

SPI - subprogram pentru îmbătrînire. Este unitate de comandă subordonată (subprogram) ce asigură efectuarea îmbătrînirii. Declanşează un ciclu de scriere sau citire, incrementează numărătoarele N_2 (adresare curentă) şi NR (numărător de rînduri); iniţializează numărătorul NR şi bistabilul SPI.

SPT - subprogram de testare. Este tot o unitate de comandă subordonată, ce conduce operaţiunile de

testare. Provoacă inițializarea numărătoarelor NC, NM, NR, incrementarea acestora și a numărătorului N_2, poziționarea pe "1" a bistabilului TESTER și inițializarea bistabilului SPT.

După cum se observă, unitățile de comandă sînt interconectate, putîndu-se apela una pe cealaltă. Principiul legăturii între două unități de comandă a fost prezentat în cap.7.

8.1.1. Blocurile funcționale ale testorului.

În acest paragraf sînt descrise, grupate pe categorii blocurile funcționale ale testorului.

Numărătoarele (fig.8.2).

N_2 - numărător de adresare curent, la nivel de capsulă. Numărătorul are douăsprezece ranguri și este sincron reversibil. Ieșirile N_2 pot fi aduse în orice stare prin introducerea informației dorite pe intrările de date, în paralel, deoarece numărătorul N_2 trebuie să poată fi încărcat cu conținutul numărătorului N_1. Incrementarea numărătorului N_2 se face cu semnalul + $1N_2$; inițializarea cu $A(o)N_2$, iar memorarea conținutului numărătorului N_1 cu comanda $(N_1) \longrightarrow N_2$.
Transportul rezultat de la ultimul rang este accesibil sub forma semnalului c_{12}.

N_1 - este al doilea numărător de adresare, are tot 12 ranguri și adresează tot la nivelul capsulelor. Acest numărător este utilizat numai în cazul parcurgerii de către testor a testului GALLOPING, în acest caz lucrînd corelat cu numărătorul N_2. Incrementarea se face cu semnalul + $1N_1$ iar inițializarea cu semnalul $A(o)N_1$. Schema bloc este prezentată în figura 8.2.

N-REG - numărător de regenerare ; generează adresa curentă de regenerare pe 6 ranguri. Incrementarea se face la semnalul GT-REG (generator tact pentru regenerare). Inițializarea este realizată de comanda STIN

NR - numărător al rîndurilor de capsule; are 4 ranguri (la modulul de 32 KO se folosesc doar rîndurile 0 și 1 care generează de fapt semnalele de adresă

$AD_{12,13}$), este incrementat la semnalul +1NR şi iniţializat prin A (o) NR.

NM – numărător al modulelor de testat.. Are două ranguri (în cele prezentate mai sus s-o arătat că numărul maxim de module ce se pot testa este 4). Avansarea numărătorului se face la comanda +1NM iar iniţializarea prin semnalul A(o)NM.

NC – numărător al ciclurilor de testare. Operatorul poate cere efectuarea a 1,2,4 sau 8 cicluri. Numărătorul are 3 ranguri. Numărătorul este incrementat de semnalul +1NC şi iniţializat de către A(o)NC.

N – Ultimul dintre numărătoare este un numărător de teste; prin conţinutul său indică ce fel de test este în execuţie (testele şi conţinutul corespunzător al numărătorului au fost prezentate deja). Numărătorul are trei ranguri. Incrementarea se face cu semnalul +1Z iar iniţializarea prin A(o)Z..

Multiplexoarele (fig.8.3)

MUX – IA – multiplexor intrare adresă. Selectează adresa din trei locuri posibile: numărătorul N_2 în cazul adresării curente, numărătorul N_1 în cazul parcurgerii testului GALLOPING sau N-hEU pe durata regenerării, şi o trimite spre intrările de adresă ale modulelor (NAD_{o-11}).
Pe primele 6 ranguri, (c.m.p.s) schema e realizată pe trei nivele, deoarece aici intervin ranguri numărătorului de regenerare. Presupunem că apare semnalul (N_1)→IA (de la generatorul de faze) de multiplexare GF-IA). Bineînţeles celelalte semnale de comandă ale multiplexorului (N_2)→IA şi LANS-REG vor fi "o". In acest caz prin porţile din primul nivel vor putea trece semnalele de la numărătorul N_1, semnalele de la numărătorul N_2 fiind blocate de "o"-ul semnalului (N_2)→IA. In al doilea nivel se găsesc circuite invertoare, pentru păstrarea în final a polarităţilor dorite. Prin nivelul trei se permite trecerea semnalelor ce vin din nivelul doi datorită validării lor de către semnalul LANS

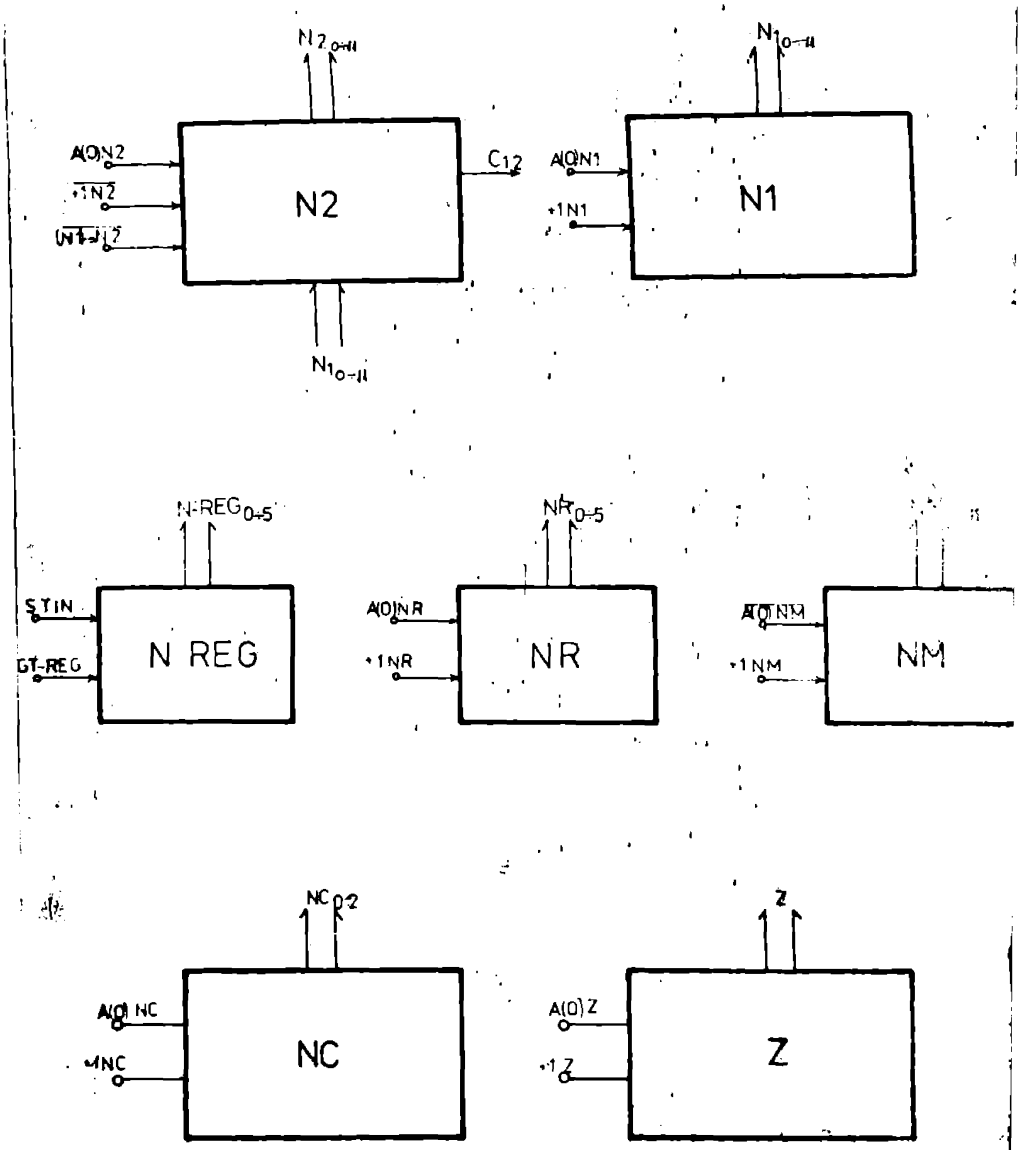

N2
A(0)N2
+1N2
N2 0-11
C12
N1
A(0)N1
+1N1
N1 0-11
N REG
S.T IN
GT-REG
N-REG 0-5
NR
A(0)NR
+1NR
NR 0-5
NM
A(0)NM
+1NM
NC
A(0)NC
+NC
NC 0-2
Z
A(0)Z
+1Z
Z

Fig. 8.2.

REG (semnalele numărătorului de regenerare sînt blocate la fel ca şi semnalele numărătorului N_2 în primul nivel, de către LANS REG "0").
Dacă de exemplu LANS REG e "1" vor fi validate doar porţile din nivelul trei, la ieşiri regăsindu-se semnalele numărătorului de regenerare N-REG.

MUX - AF - sistemul de vizualizare permite urmărirea informaţiei din patru puncte:
- date intrare
- date ieşire
- adresa curentă
- rezultatul comparării bit cu bit.

Semnalele ce determină vizualizarea uneia sau alteia din mărimi sînt nivelele de tensiune (de la generatorul de faze afişare GF-AF; A-ID; A-ED; A-IA; A-COMP.
Funcţionarea este analogă cu cea a multiplexorului de intrare adresă: nu sînt validate decît intrările a căror selecţie este "1".

MUX - ID - multiplexorul de intrare date. Datele ce urmează a fi memorate într-un modul pot proveni de la:
- cheile de date de la panoul de comandă;
- numărătorul N_2 ;
- bistabilul D (poziţionat pe "0" sau "1");

Acest multiplexor selectează una din aceste 3 surse de date.

Generatoare de faze pentru multiplexoare (Fig.8.4)

GF - AF - generator de faze pentru multiplexoare afişare. Are rolul de a transforma impulsurile a-id; a-ed; a-comp; a-ia venite de la unităţile de comandă, în nivele de tensiune cu care să atace multiplexorul de afişare. Ca şi celelalte generatoare de faze (pe care le vom descrie în continuare) generatorul de faze pentru afişare conţine un codificator, un registru tampon şi un decodificator propriu.
Registrul tampon este realizat cu două bistabile

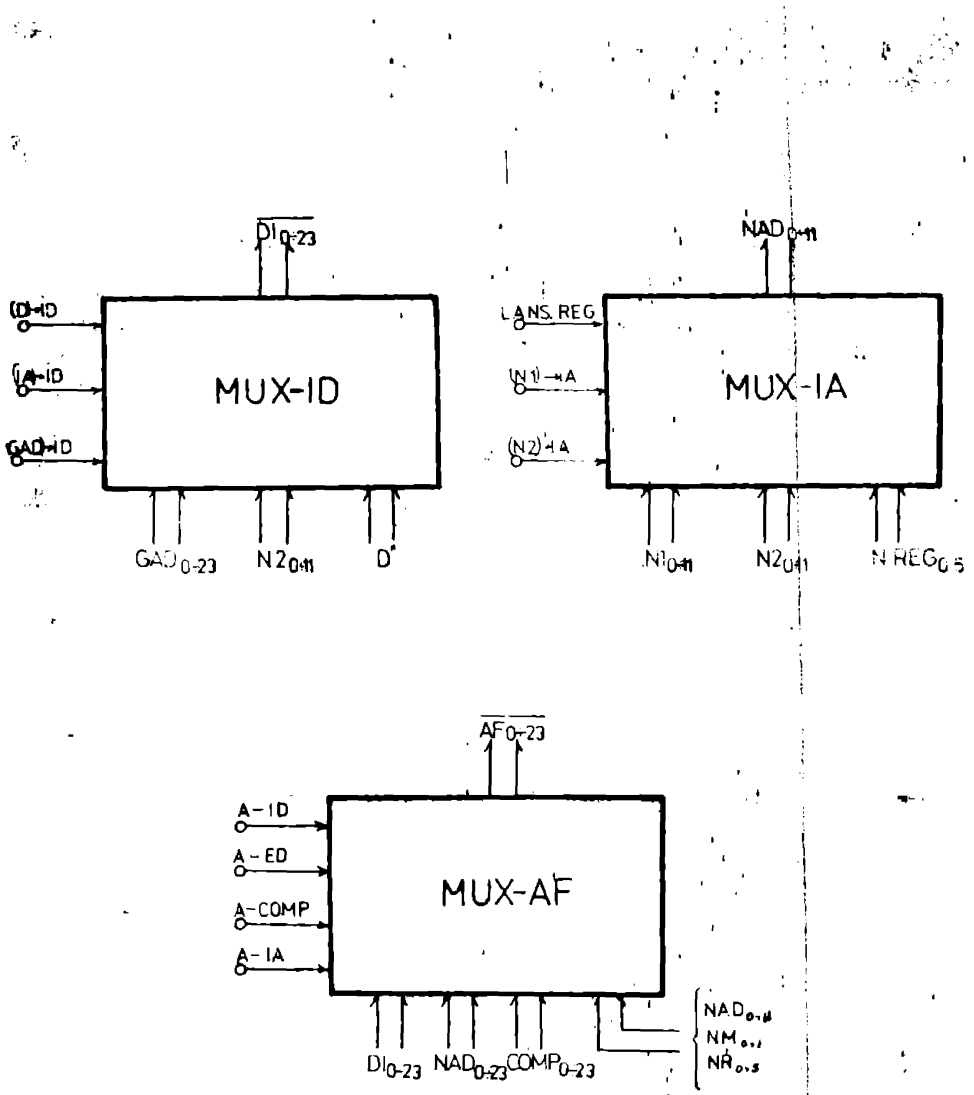

DI0-23
(D)→ID
(A)→ID
GAD→ID
MUX-ID
GAD0-23
N20-11
D
NAD0-11
LANS. REG.
(N1)→A
(N2)→A
MUX-IA
N10-11
N20-11
N REG0-5
AF0-23
A-ID
A-ED
A-COMP
A-IA
MUX-AF
DI0-23
NAD0-23
COMP0-23
NAD0-11
NM0-1
NR0-5

Fig. 8.9.

RS. Pentru p oiectare întocmim următorul tabel:

a-id	a-ed	a-comp	a-ia	B_o	B_1	A-ID	A-ED	A-COMP	A-IA
1				o	o	1			
	1			o	1		1		
		1		1	o			1	
			1	1	1				1

De aici putem scrie ecuaţiile pentru intrările bistabilelor:

$\bar{r}_o = \overline{a\text{-}id + a\text{-}ed} = \overline{a\text{-}id}.\overline{a\text{-}ed}$

$\bar{s}_o = \overline{a\text{-}comp + a\text{-}ia} = \overline{a\text{-}comp}.\overline{a\text{-}ia}$

$\bar{r}_1 = \overline{a\text{-}id + a\text{-}comp} = \overline{a\text{-}id}.\overline{a\text{-}comp}$

$\bar{s}_1 = \overline{a\text{-}ed + a\text{-}ia} = \overline{a\text{-}ed}.\overline{a\text{-}ia}$

Si ecuaţiile nivelelor de tensiune de la ieşire:

$A\text{-}ID = \bar{B}_o.\bar{B}_1$

$A\text{-}ED = \bar{B}_o.B_1$

$A\text{-}COMP = B_o.\bar{B}_1$

$A\text{-}IA = B_o.B_1$

F-ID – generator de faze pentru multiplexare intrare date Transformă impulsurile (d)→id; (ia)→id; (gad)→id, în nivele necesare etacării multiplexorului de intrare date. Întocmim tabelul pentru scrierea ecuaţiilor:

(d)→id	(ia)→id	(gad)→id	B_o	B_1	(D)→ID	(IA)→ID	(GAD)→ID
1			o	o	1		
	1		o	1		1	
		1	1	o			1

$\bar{r}_o = \overline{(d)\to id + (ia)\to id} = \overline{(d)\to id}.\overline{(ia)\to id}$

$\bar{s}_o = \overline{(gad)\to id}$

$\bar{r}_1 = \overline{(d)\to id + (gad)\to id} = \overline{(d)\to id}.\overline{(gad)\to id}$

$\bar{s}_o = \overline{(ia)\to id}$

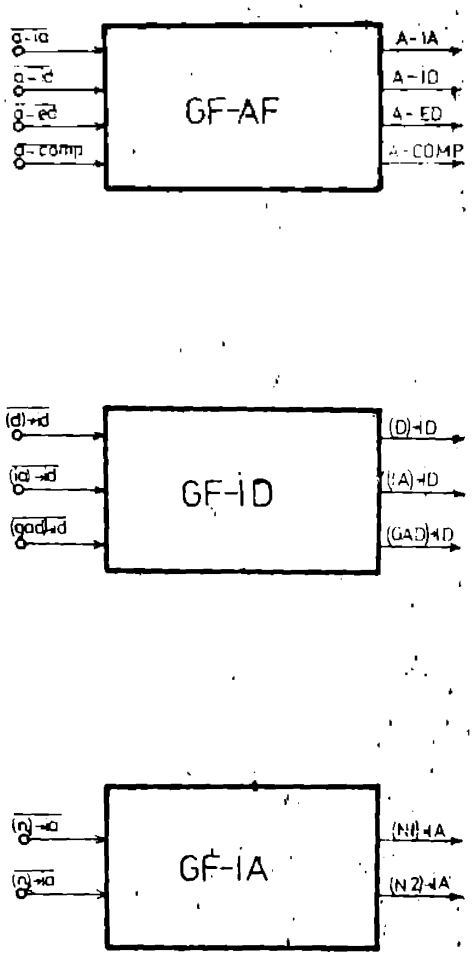


Fig. 8.4.

$(D) \longrightarrow ID = \overline{B}_o \cdot \overline{B}_1$

$(IA) \rightarrow ID = \overline{B}_o \cdot B_1$

$(GAD) \longrightarrow ID = B_o \cdot \overline{B}_1$

GF - IA - generator de faze pentru multiplexare intrare adresă. Transformă impulsurile $(n_1) \rightarrow$ ia şi $(n_2) \longrightarrow$ ia în nivele de tensiune necesare multiplexorului intrare adresă. Este cel mai simplu dintre generatoare; cele două impulsuri atacă direct intrările bistabilului (ce conatituie registrul tampon) iar ieşirile bistabilului sînt tocmai nivelele necesare.

Cocificatoarele (fig.8.5)

COD - NCF - codificatorul numărului de cicluri final. Numărul de cicluri de testare (1,2,4 sau 8) este fixat de către operator prin intermediul a 4 chei de pe panoul de comandă. Acest număr este cocificat prin doi biţi:

Kc_1	Kc_2	Kc_4	Kc_8	NCF_1	NCF_o
1				o	o
	1			o	1
		1		1	o
			1	1	1

$NCF_o = Kc_2 + Kc_8 = \overline{\overline{Kc}_2 \cdot \overline{Kc}_8}$

$NCF_1 = Kc_4 + Kc_8 = \overline{\overline{Kc}_4 \cdot \overline{Kc}_8}$

COD - NMF - codificatorul numărului de module final. Numărul de module de testat nu este fixat de la chei, de pe panou ci e dat de către nişte senzoare electro-mecanice (KM_i). Numărul de module se codifică tot pe doi biţi (NMF_o ; NMF_1) :

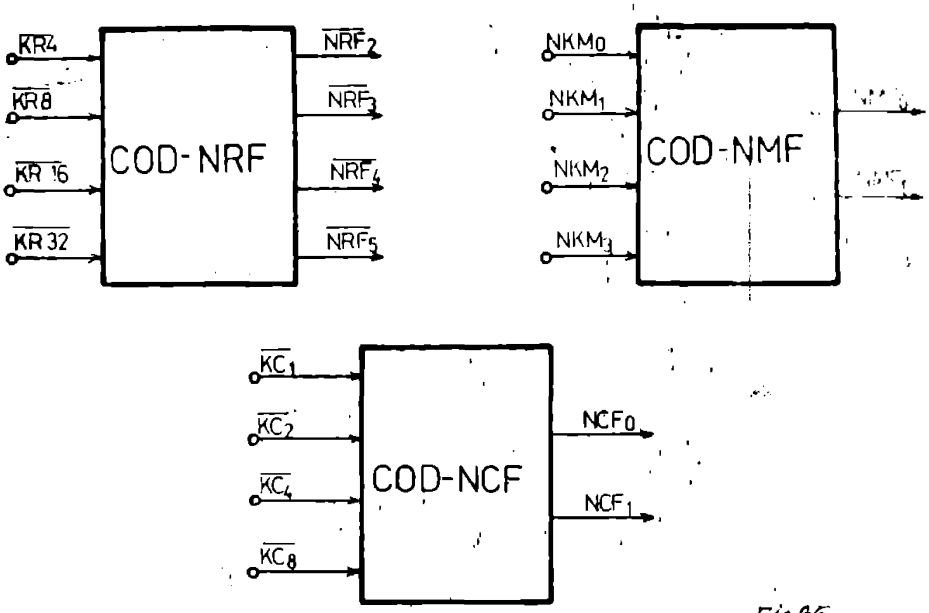
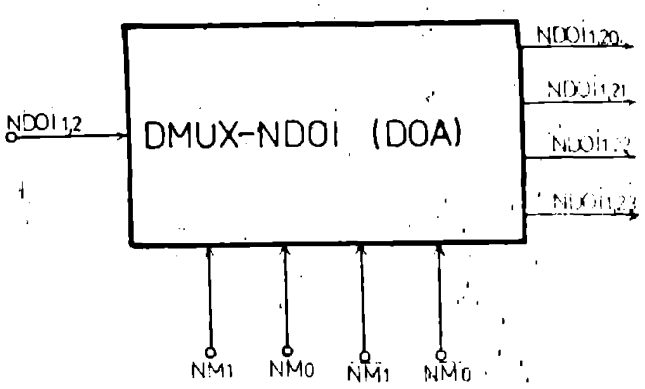
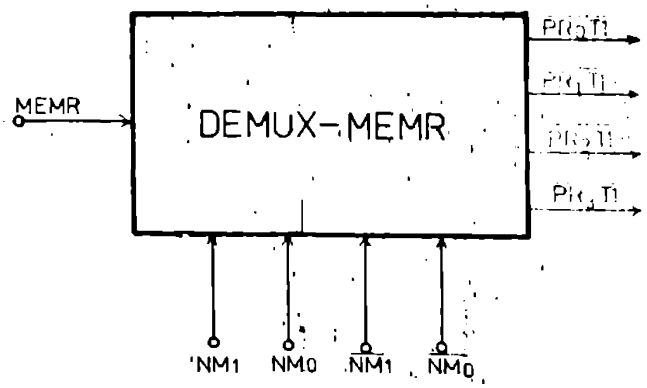

KR4
KR8
KR 16
KR 32
COD-NRF
NRF2
NRF3
NRF4
NRF5
NKM0
NKM1
NKM2
NKM3
COD-NMF
KC1
KC2
KC4
KC8
COD-NCF
NCF0
NCF1

Fig 65.


NDOI1,2
DMUX-NDOI (DOA)
NDOI1,20
NDOI1,21
NDOI1,22
NDOI1,23
NM1
NM0
NM1
NM0


MEMR
DEMUX-MEMR
NM1
NM0
NM1
NM0

KM_0	KM_1	KM_2	KM_3	NMF_1	NMF_0
1				0	0
1	1			0	1
1	1	1		1	0
1	1	1	1	1	1

$NMF_1 = KM_2$

$NMF_0 = KM_3 + KM_1 \cdot \overline{KM_2}$

COD - NRF - codificator al capacității modulelor. Din exterior această capacitate este fixată de către operator prin intermediul a 4 chei de pe panoul de comandă (8 K, 16 K, 32 K, 64 K). Capacitatea modulelor este codificată sub forma a 4 biți ce reprezintă capacitatea în număr de rînduri de capsule:

KR_4	KR_8	KR_{16}	KR_{32}	NRF_5	NRF_4	NRF_3	NRF_2	NRF_1	NRF_0
1							1		
	1					1			
		1			1				
			1	1					

$NRF_0 = 0$

$NRF_1 = 0$

$NRF_2 = KR_4$

$NRF_3 = KR_8$

$NRF_4 = KR_{16}$

$NRF_5 = KR_{32}$

Demultiplexoarele (fig.6.6)

DEMUX-DOA - are rolul:de a trimite semnalul $DOA(NDOI_{1,2})$, spre modulul curent. Are la intrări ieșirile numărătorului de module NM.

DEMUX-MEMER-în cazul apariției unei defecțiuni trebuie să se știe la care dintre module a survenit, prin poziționarea pe "1" a biestabilului asociat, din tamponul T_1. Acest lucru e făcut de către unul dintre

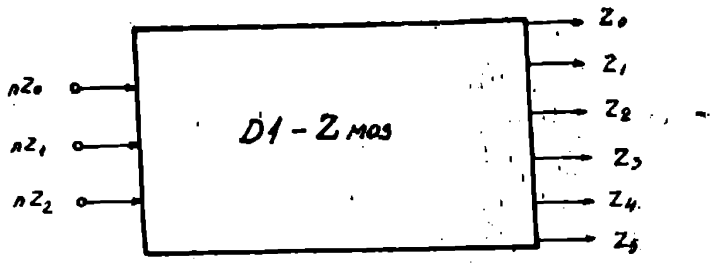


Fig. 8.7.

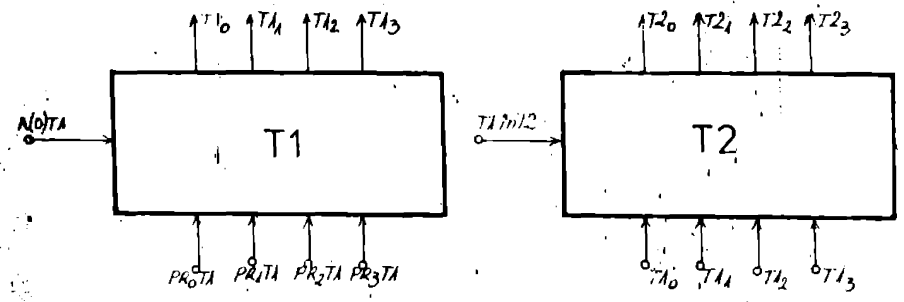


Fig. 8.8.a

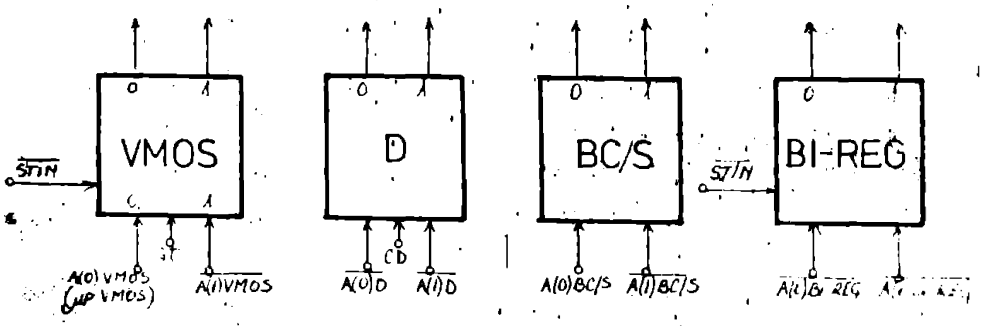


Fig. 8.8.b

semnalele PR_iT_1 de la ieşirea acestui demultiplexor. Deci semnalul MEMER e demultiplexat spre modulul curent (la care a survenit eroarea).

Decodificatoare (fig.6.7)

D_1-Z_{MOS} - în afară de decodificatoarele înglobate în schemele generatoarelor de faze pentru multiplexare, mai există acest decodificator, care decodifică starea curentă a numărătorului de teste Z, generînd semnalele Z_1 Z_5.

Registre tampon (fig.6.8)

T_1 - după cum am arătat, dacă apare o defectiune, trebuie să se ştie la care dintre module a intervenit. Tamponul T_1 conţine 4 bistabile, cîte unul asociat fiecărui modul. Semnalul de memorare a erorii MEMER este demultiplexat spre cele 4 bistabile sub forma semnalelor PR_iT_1, bistabilul asociat modulului la care a survenit eroarea este poziţionat de respectivul semnal pe "1".

T_2 - numărul modulului la care a intervenit greşeala trebuie trimis spre microprocesor. Acest lucru se realizează cu ajutorul acestui tampon, care ale cărui ieşirile conectate prin intermediul unor invertoare cu trei stări la magistrala de date a microprocesorului. Transferul conţinutului tamponului T_1 în tamponul T_2 se face la comanda T_1 în T_2. Această comandă se dă întotdeauna după ce unităţile SPT sau UC-GAL şi-au terminat activităţile, ele fiind singurele ce pot detecta o eroare.

Bistabile (fig.6.8)

D - se poziţionează pe "1" sau pe "0", dacă datele ce trebuiesc înscrise în modul sînt 11...11 sau 00...00. Are intrările de forţare pe "1" şi pe "0" conectate la semnalele A(1)D ; A(0)D. Tactul bistabilului este format printr-o poartă SI ce are la intrări tactul de la sistemul cu microprocesor (10 MHz) şi comanda de complementare date CD.

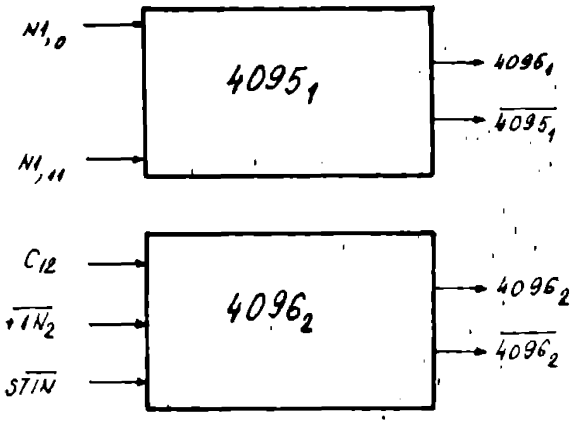


Fig. 8.9.

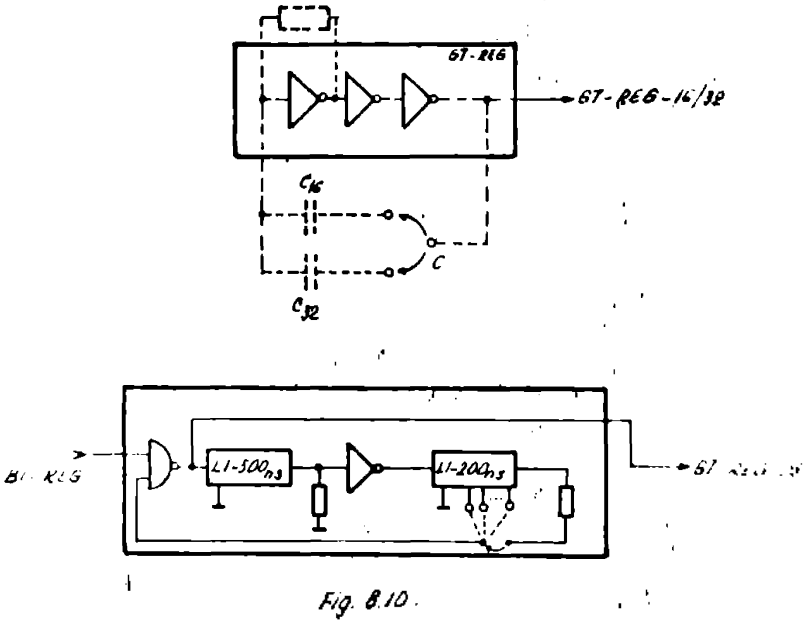


Fig. 8.10.

V_{MOS} - cînd se execută transferul (T_1) T_2 acest bistabil e poziţionat pe "1" şi dă întrerupere la microprocesor. După ce microprocesorul serveşte întreruperea, readuce la "o" bistabilul V_{MOS} şi permite astfel plecarea mai departe a testorului. Bistabilul V_{MOS} este format de fapt din două bistabile. Este iniţializat de semnalul $\overline{STIN}$, adus la "1" cu comanda $\overline{A(1)V_{MOS}}$. Este adus la "o" de către microprocesor cu comanda $A(o)V_{MOS}$(sau $\overline{V_{MOS}}$).

BC/S - bistabil de citire - scriere; cînd se află poziţionat pe "1", indică o operaţie de citire, iar cînd conţinutul său e "o", se execută o scriere. Este realizat cu un bistabil JK ce are intrarea J la "1" logic. La intrarea K e conectat semnalul STIN(stare iniţială). Are intrarea de forţare pe "1" conectată la semnalul A(1)BC/S.

BI-REG - bistabilul de îmbătrînire-regenerare. Memorează o cerere de îmbătrînire-regenerare. E adus la "1" de către semnalul A(1)BI-REG, iar la "o" de către semnalul A(o)BI-REG sau STIN.

Sesizoare conţinut numărătoare (fig.8.9)

$4o95_1$ - determină momentul cînd numărătorul N_1 a ajuns la capacitatea maximă (este "plin" de "1"). Dacă toate rangurile numărătorului sînt "1", la ieşire va apare semnalul ($4o95_1$).

$4o96_2$ - determină momentul cînd numărătorul N_2 face tranziţia de la 11...11 la oo...oo. Este realizat sub formă de bistabil. Iniţializarea lui se face cu semnalul STIN. Semnalul c_{12} de la intrare este transportul dinspre ultimul rang.

Generatoare de tact (fig.8.10)

GT-AUT - este generatorul de tact automat. Are o frecvenţă de 10 MHz şi este conţinut în cadrul sistemului de calcul cu microprocesor.

GT-REG-16/32-generator de tact pentru regenerare. Generează un tact cu perioada de 16 sau 32 μs (în funcţie de conexiunile $c-c_{16}$ sau $c-c_{32}$). Este realizat cu 3 porţi invertoare.

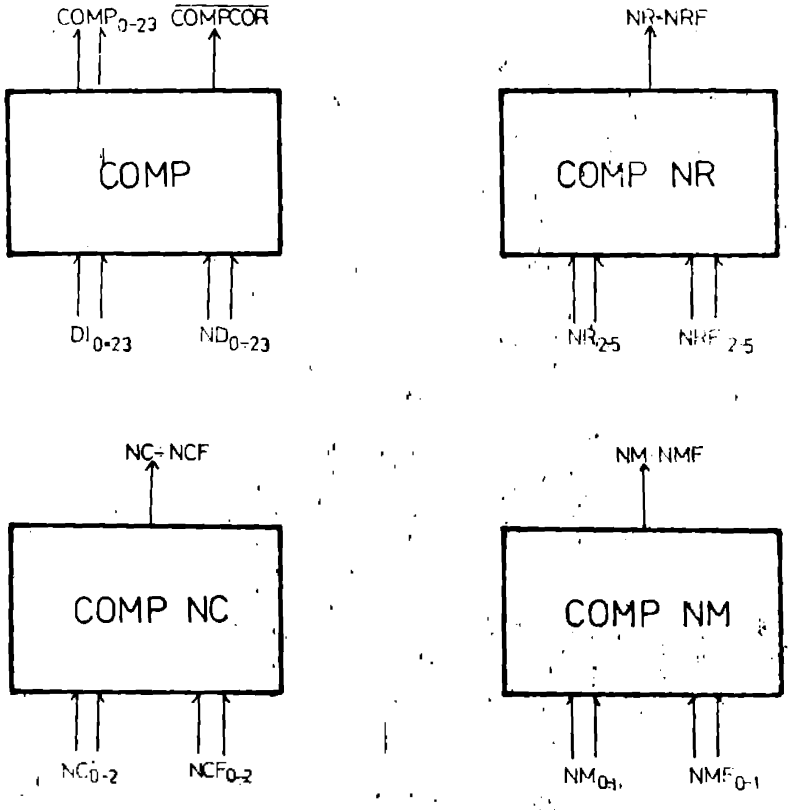

COMP0-23
COMPCOR
COMP
DI0-23
ND0-23
NR-NRF
COMP NR
NR2-5
NRF2-5
NC-NCF
COMP NC
NC0-2
NCF0-2
NM-NMF
COMP NM
NM0-1
NMF0-1

Fig. 8.11.

GT-REG-05 - generator de tact pentru îmbătrînirea de regenerare, dacă bistabilul BI-REG este în starea "1". Este realizat cu două linii de întîrziere.

Comparatoare (fig.8.11)

COMP - compară pe 16 rânduri, bit cu bit, datele de intrare în module (datele ce trebuie memorate) cu cele de ieşire (datele ce s-au memorat şi s-au citit). La comparare cu rezultat corect apare semnalul COMPCOR = "1".

COMP-NR - compară numărul de rînduri de capsule curent, cu cel final, specificat de către COD-NRF, generînd în caz de coincidenţă semnalul NR=NRF. Semnalul NR=NRF ne semnalizează ajungerea la ultimul rînd de capsule.

COMP-NM - compară numărul de module curent (dat de numărătorul de module, NM) cu numărul de module final dat de codificatorul numărului de module final, COD-NMF. În caz de coincidenţă se generează semnalul NM=NMF.

COMP-NC - compară conţinutul numărătorului de cicluri, NR, cu ieşirile codificatorului numărului de cicluri final COD-NCF. Dacă la celelalte comparatoare nu a fost necesară scrierea ecuaţiilor pentru sintetizare, aici deoarece avem de comparat un număr du trei rânduri (conţinutul numărătorului de cicluri),cu unul cu două rânduri (ieşirea codificatorului numărului de cicluri). De aceea vom codifica conţinutul numărătorului de cicluri pe doi biţi:

NC			NC'	
NC_2	NC_1	NC_0	NC'_1	NC'_0
0	0	0	0	0
0	0	1	0	1
0	1	1	1	0
1	1	1	1	1

$NC'_1 = NC_1$

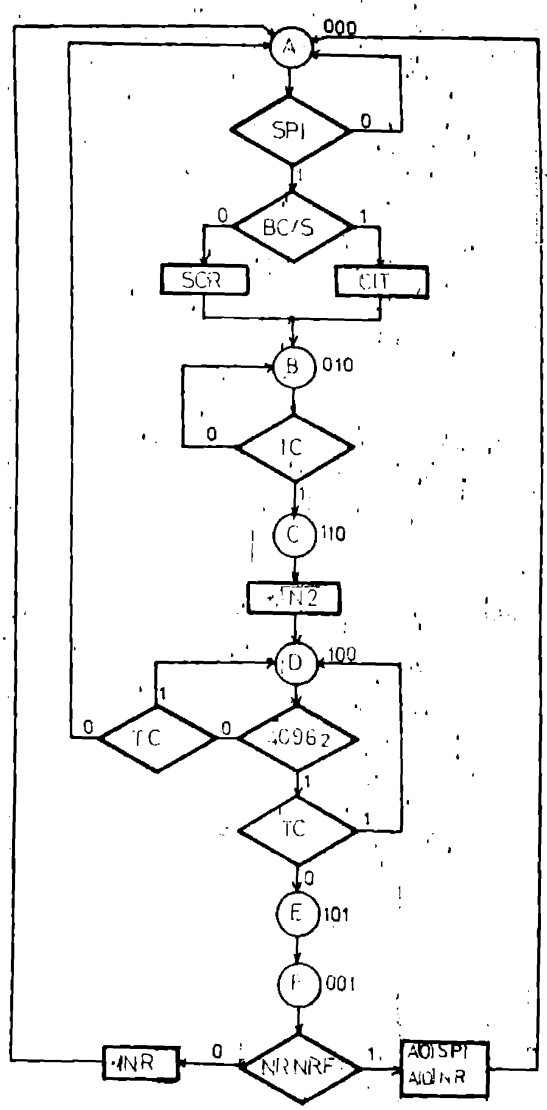

000
A
SPI
0
1
BC/S
0
1
SCR
CIT
B
010
IC
0
1
C
110
N2
D
100
1
TC
0
0
C962
1
TC
1
0
E
101
F
001
INR
0
NRNRF
1
AO SPI
AO INR

Fig. 8.12.

$NC_0'=\overline{NC}_2.\overline{NC}_1.NC_0+NC_2.NC_1, NC_0=NC_0(\overline{NC}_2.\overline{NC}_1+NC_2.NC_1)$

Acum compararea se face ca şi la celelalte comparatoare prin circuite SAU-EXCLUSIV.

8.1.2. Proiectarea structurii de comandă în detaliu

Proiectarea SPI

SPI - este unitatea de comandă ce conduce operaţiunile de îmbătrînire (subprogram de îmbătrînire). Apelarea acestei unităţi de comandă se face prin poziţionarea pe "1" a bistabilului asociat SPI (cît timp bistabilul SPI este "o", unitatea de comandă SPI stă în bucla de aşteptare).

Apelată, unitatea verifică dacă se cere îmbătrînire la citire sau la scriere (bistabilul citire/scriere BC/S este "1" sau "o"), pentru a se genera comenzile respective (SCR dacă BC/S este "o" şi CIT dacă BC/S e "1"). In continuare operaţiunile sînt identice, indiferent dacă se face îmbătrînirea la scriere sau la citire: se aşteaptă scurgerea timpului corespunzător duratei unui ciclu (TC) după care se formează următoarea adresă prin incrementarea numărătorului de adresare curentă ($+1N_2$). Se verifică dacă acest numărător a ajuns la capacitatea maximă plus o unitate (deci oo...o). Verificarea se face prin senzorul de capacitate maximă $4o96_2$.

In cazul în care nu s-a ajuns la capacitatea maximă după scurgerea duratei unui ciclu (TC=o) se revine în starea iniţială, procedîndu-se la citirea (CIT) respectiv înscrierea (SCR) la noua adresă.

Dacă s-a ajuns la capacitatea maximă, se aşteaptă şi-n acest caz terminarea ciclului, după care se verifică dacă au fost parcurse toate rîndurile de capsule (NR=NRF). Dacă nu s-a parcurs ultimul rînd, este incrementat numărătorul de rînduri ($+1NR$) pentru trecerea le următorul rînd. Dacă ultimul rînd parcurs a fost şi cel din urmă, este iniţializat numărătorul de rînduri, şi bistabilul SPI, astfel unitatea de comandă SPI şi-a încheiat activitatea intrînd în bucla de aşteptare iniţială şi predă comanda altei unităţi.

Ordinograma de funcţionare a SPI se găseşte în figura 8.12. Cele 6 stări se codifică cu ajutorul a trei bistabile. Diagrama stărilor curente o prezentăm în figura 8.13 iar

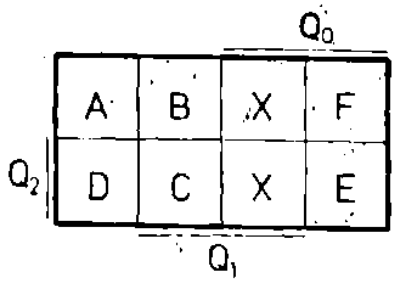


Fig. 8.13

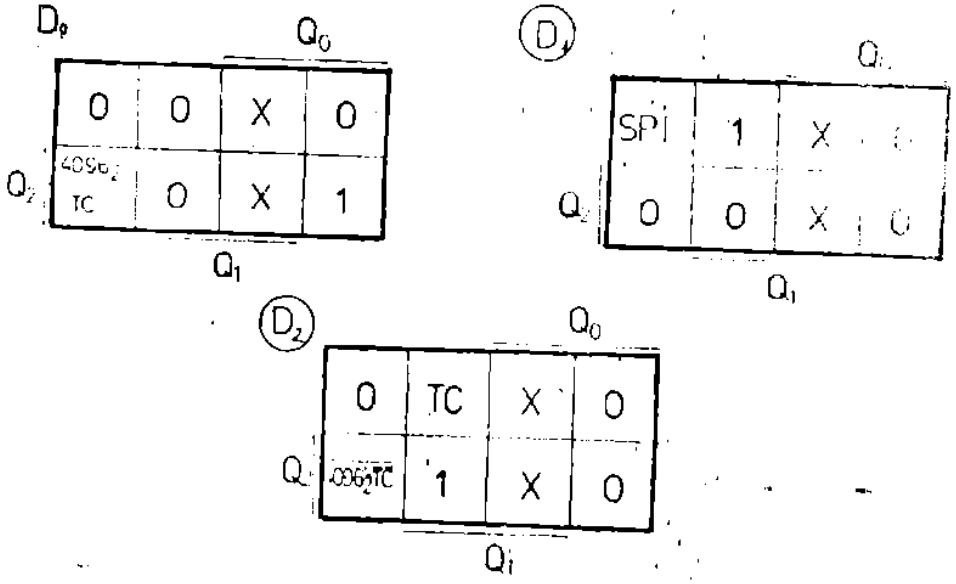


Fig. 8.14

cea a noilor stări în figura 8.14.

Ecuaţiile de intrare pentru cele 3 bistabile ale registru lui de stări sînt:

$$D_0 = Q_0.Q_2 + 4o96_2.\overline{TC}.\overline{Q}_1.Q_2$$

$$\overline{D}_0 = \overline{Q_0.Q_2}.4o96_2\ \overline{TC}.\overline{Q}_1.Q_2$$

$$\overline{D}_1 = Q_2 + Q_0 + \overline{SPI}.\overline{Q}_1$$

$$D_1 = \overline{Q}_0.\overline{Q}_2.\overline{Q_1}.\overline{SPI}$$

$$\overline{D}_2 = \overline{Q}_0 + \overline{Q}_1.\overline{Q}_2 + \overline{TC}.\overline{Q}_3 + \overline{4o96_2}.\overline{TC}.\overline{Q}_1$$

$$D_2 = \overline{Q}_0.\overline{Q}_1.\overline{Q}_2.\overline{TC}.\overline{Q}_2.\overline{4o96}_2.\overline{TC}.\overline{Q}_1$$

Ecuaţiile au fost aduse la forma de produse logice pentru ca sintetizarea să se facă cu circuite ŞI-NU.

Ecuaţiile funcţiilor de ieşire sînt:

$$SCR = A.SPI.\overline{BC/S} = \overline{Q}_0.\overline{Q}_1.\overline{Q}_2.SPI.\overline{BC/S}$$

$$CIT = A.SPI.BC/S = \overline{Q}_0.\overline{Q}_1.\overline{Q}_2.SPI.BC/S$$

$$+1N_2 = c = \overline{Q}_0.Q_1.Q_2$$

$$+1NR = F.\overline{NR=NRF} = Q_0.\overline{Q}_1.\overline{Q}_2.\overline{NR = NRF}$$

$$A(o)SPI = A(o)NR=F.\overline{NR=NRF}=Q_0.\overline{Q}_1.\overline{Q}_2.\overline{NR=NRF}$$

Proiectarea UC-CODAR

UC-CODAR - este unitatea de comandă, generatoare a codului de eroare. Bistabilul asociat acestei unităţi este TESTAR. Apelarea unităţii se realizează prin poziţionarea acestui bistabil pe "1" (cît timp TESTAR e "o" unitatea UC-CODAR rămîne în bucla de aşteptare iniţială).

Odată apelată, UC-CODER dă un semnal de citire (CIT) şi după scurgerea timpului de acces (TACC="1"), se verifică dacă informaţia citită este corectă (dacă semnalul COMPCOR(ieşirea comparatorului dintre datele citite şi cele înscrise) este "1") Dacă înscrierea s-a făcut corect (deci COMPCOR este "1") se iniţializează bistabilul TESTAR, adică UC-CODAR intră în bucla de aşteptare iniţială, comanda putînd fi preluată de către altă unitate de comandă.

În cazul că s-a detectat o eroare ($\overline{COMCOR}$ e "o"), se veri-

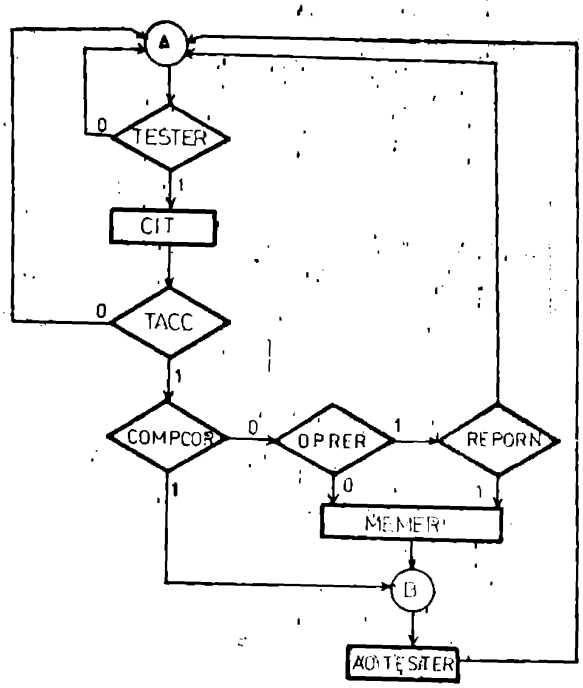
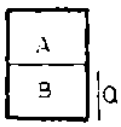


Fig. 8 15.



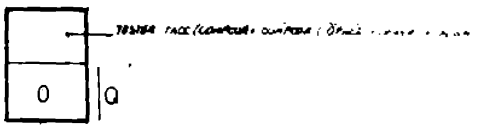


Fig. 8.16.

fică dacă de la panou s-a cerut oprirea în caz de eroare (OPRER e "1"). Dacă nu s-a cerut oprirea (OPRER e "o"), UC-CODER generează semnalul de memorare a erorii MEMER, după care predă controlul (A(o) TESTER). In cazul că s-a cerut oprirea testorului în cazul unei erori, UC-CODER intră într-o buclă de aşteptare, pînă la apariţia semnalului REPORN (repornire de la panou). Odată cu apariţia acestui semnal se generează acelaşi semnal MEMER, după care UC-CODER intră în bucla de aşteptare iniţială.

Organigrama de funcţionare este prezentată în figura 8.15 iar în figura 8.16 diagrama stărilor curente, şi diagrama noilor stări.

Ecuaţia de intrare pentru bistabilul ce formează registrul stărilor:

$$D_0=\overline{Q}_0.TESTER.TACC.\overline{COMPCOR}.\overline{Q}_0.TESTER.TACC.\overline{COMPCOR}.OPRER.$$
$$\overline{Q}_0.TESTER.TACC.\overline{COMPCOR}.OPRER.REPORN$$

Ecuaţiile funcţiilor de ieşire:

$$CIT = A.TESTER = \overline{Q}_0.TESTER$$

$$A(o)TESTER = B = Q_0$$

$$MEMER = \overline{Q}_0.TESTER.TACC.\overline{COMPCOR}.\overline{OPRER}+\overline{Q}_0.TESTER.TACC.$$
$$\overline{COMPCOR}.OPRER.REPORN$$

Proiectarea SPI

SPT - este unitatea de comandă ce conduce acţiunea de testare (SPI - subprogram de testare). Bistabilul asociat acestei unităţi este SPT. Apelarea unităţii SPT se face prin poziţionarea pe "1" a acestui bistabil. (Dacă bistabilul SPT e "o" unitatea va rămîne în buclă de aşteptare).

După apelare, primele operaţiuni executate de unitatea SPI sînt iniţializarea numărătoarelor de cicluri (NC) de module (NM) şi de rînduri (NR). Se predă apoi comanda altei unităţi de comandă, şi anume UC-CODER, unitatea de generare a codului de eroare, prin poziţionarea bistabilului asociat acesteia (TESTER) pe "1".

După cum s-a arătat în paragraful precedent, UC-CODER

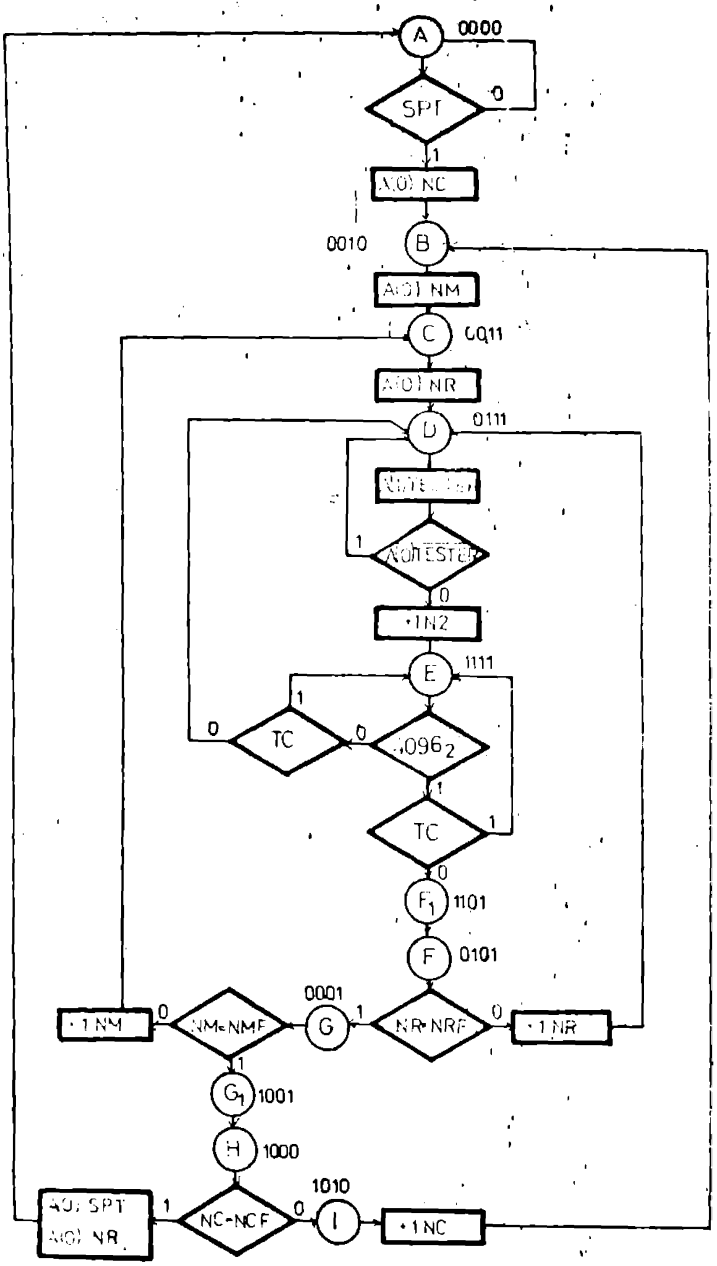

A
0000
SPT
0
1
A(0) NC
0010
B
A(0) NM
C
0011
A(0) NR
D
0111
1
0
•1N2
E
1111
TC
0
1
$,096_2$
TC
F1
1101
F
0101
0001
•1 NM
NM<NMF
G
NR<NRF
•1 NR
G1
1001
H
1000
1010
A(0) SPT
A(0) NR
NC<NCF
I
•1 NC

Fig. 8.17.

iniţiază o citire urmată de comparare şi eventual de generarea codului de eroare. In acest timp unitatea SPT stă într-o buclă de aşteptare, în care rămîne pînă cînd unitatea apelată UC-CODER îşi termină activitatea (fapt semnalat de A(o) IESIER, ce anulează bistabilul IESIER, determinînd pe UC-CODER să intre în buclă de aşteptare. După ce preia din nou conducerea, unitatea SPI pregăteşte noua adresă prin incrementarea numărătorului de adresare curentă ($+1N_2$). Se verifică apoi dacă acest numărător a ajuns la valoarea maximă plus 1(oo...o) adică s-au epuizat toate adresele dintr-un rînd de capsule. Dacă nu s-au epuizat după scurgerea duratei unui ciclu, se apelează din nou unitatea UC-CODER, care verifică corectitudinea înscrierii la noua adresă. In cazul că a ajuns la adresa maximă (4096_2 este "1"), după scurgerea duratei unui ciclu, se verifică dacă s-a ajuns la ultimul rînd de capsule (NR=NRF). Dacă nu, se trece la următorul rînd prin incrementarea numărătorului de rînduri (+1NR) procedîndu-se ca şi mai înainte la apelarea unităţii UC-CODER. După ce s-a ajuns la ultimul rînd (NR=NRF este "1") se verifică dacă modulele s-au epuizat (NM=NMF). Dacă nu,se trece la următorul modul prin incrementarea numărătorului de module (+1NM), se iniţializează numărătorul de rînduri (A(o)NR) pentru a se începe cu primul rînd din noul modul şi se apelează UC-CODER. Dacă şi modulele s-au epuizat, se verifică dacă s-au executat atîtea cicluri de testare cîte au fost fixate de la panou (NC=NCF). Dacă mai sînt cicluri de executat, se incrementează numărătorul de cicluri (+1NC), se iniţializează numărătorul de module şi rînduri, procedîndu-se la un ciclu identic de testare. Dacă însă şi numărul de cicluri s-a epuizat (NC= NCF este "1"), se iniţializează numărătorul de rînduri (A(o)NR) şi prin poziţionarea pe "o" a bistabilului SPI, unitatea SPI îşi încheie activitatea, intrînd în buclă de aşteptare iniţială.

Organigrama de funcţionare este prezentată în *fig.*8.17, diagrama stărilor curente în fig.8.18 iar diagramele stărilor următoare, în *fig.*8.19.

Ecuaţiile de intrare pentru cele 4 bistabile ale registrului stării sînt:

$$D_o = Q_3.\overline{Q}_2+\overline{Q}_o.\overline{Q}_1$$

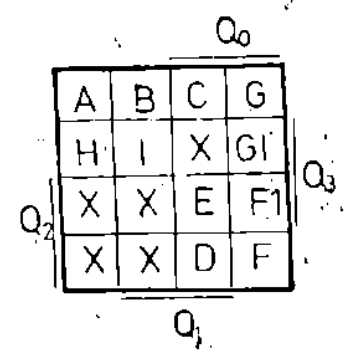


Fig. 8.18

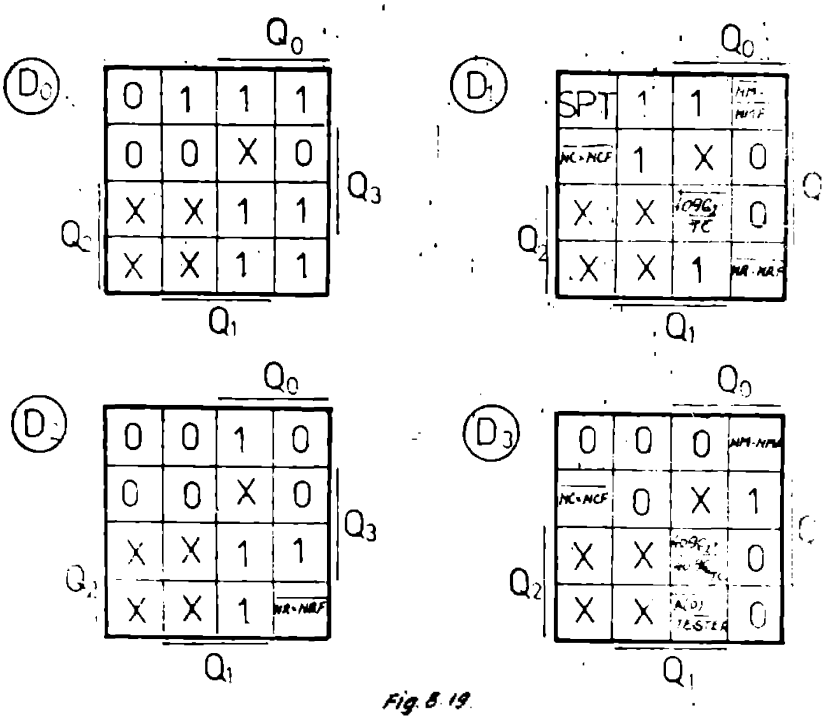


Fig. 8.19

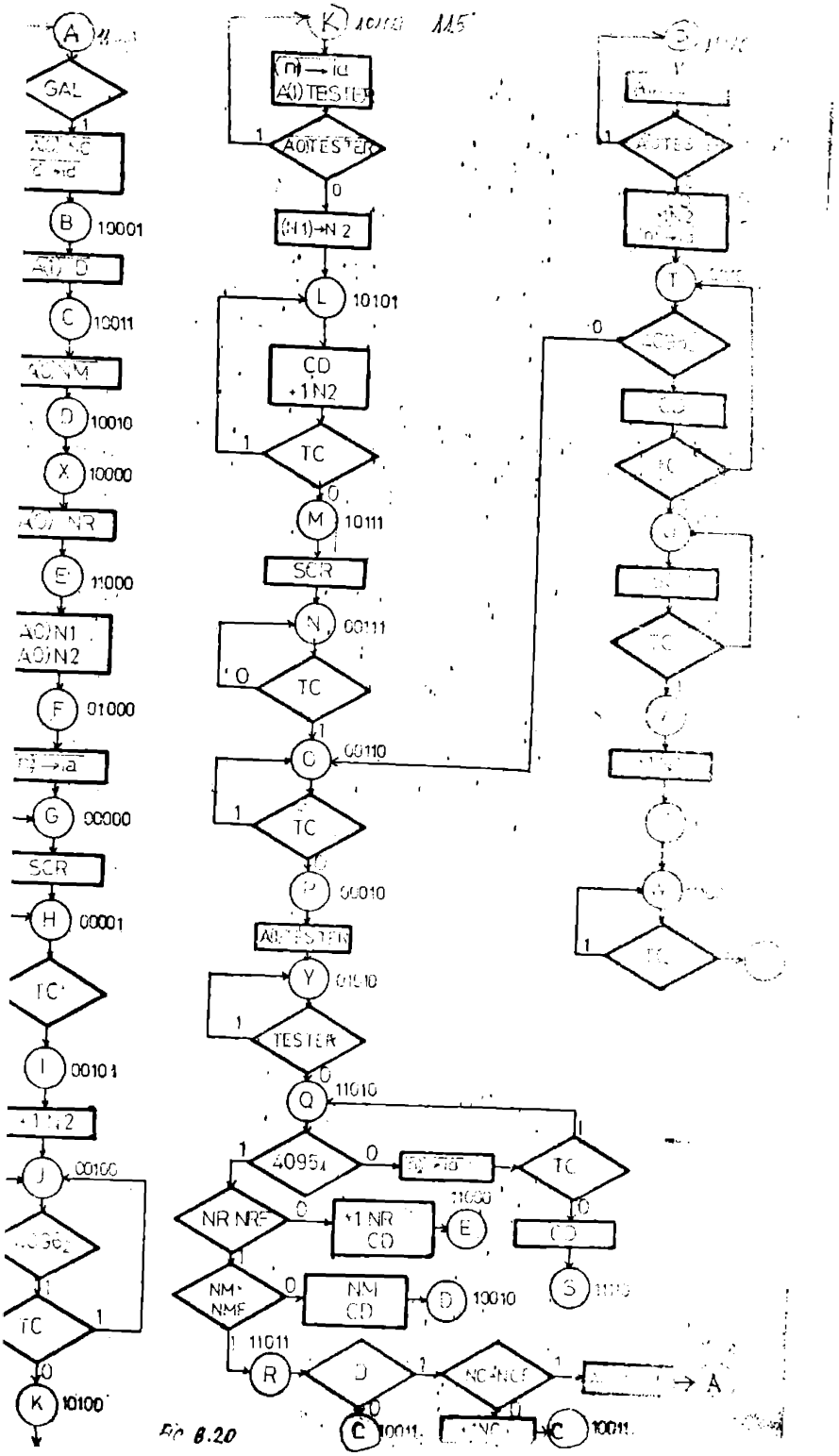

A
GAL
B
10001
C
10011
D
10010
X
10000
E
11000
AO)N1
AO)N2
F
01000
G
00000
SCR
H
00001
TC
I
00101
J
00100
TC
K
10100
K
(N)→ID
A(I) TESTER
A(I)TESTER
(N1)→N 2
L
10101
CD
+1N2
TC
M
10111
SCR
N
00111
TC
C
00110
TC
P
00010
A(I)TESTER
Y
01010
TESTER
Q
11010
409X
NR NRF
+1 NR
CD
E
11000
NM
NMF
NM
CD
D
10010
R
11011
D
NC NCF
C
10011
NC
C
10011
TC
CD
S
A

Fig 8.20

$$D_0 = \overline{Q_3 \cdot \bar{Q}_2 \cdot \bar{Q}_0 \cdot \bar{Q}_1}$$

$$D_1 = Q_1 \cdot \bar{Q}_2 + Q_1 \cdot \bar{Q}_3 + \overline{4096_2 \cdot \overline{TC}} \cdot Q_1 + \overline{NR=NRF} \cdot Q_2 \cdot \bar{Q}_3 + \overline{NM=NMF} \cdot Q_0 \cdot Q_2 \cdot \bar{Q}_3 +$$
$$SPT \cdot \bar{Q}_0 \cdot \bar{Q}_3 + \overline{NC=NCF} \cdot \bar{Q}_0 \cdot Q_3$$

$$D_2 = Q_0 \cdot Q_1 + Q_3 \cdot Q_2 + \overline{NR=NRF} \cdot Q_2$$

$$D_2 = \overline{Q_0 \cdot Q_1} \cdot \overline{Q_3 \cdot Q_2} \cdot \overline{NR=NRF} \cdot Q_2$$

$$D_3 = Q_0 \cdot \bar{Q}_2 \cdot Q_3 + NM=NMF \cdot Q_0 \cdot \bar{Q}_1 \cdot Q_2 + \overline{NC=NCF} \cdot \bar{Q}_0 \cdot Q_3 \cdot Q_1 + (4096_2 + 4096_1 \cdot \overline{TC}) \cdot$$
$$\cdot Q_1 \cdot Q_2 \cdot Q_3 + A(0)TESTER \cdot Q_1 \cdot Q_2 \cdot \bar{Q}_3$$

$$D_3 = Q_0 \cdot \bar{Q}_2 \cdot Q_3 \cdot \overline{NM=NMF} \cdot Q_0 \cdot \bar{Q}_1 \cdot Q_2 \cdot \overline{NC=NCF} \cdot \bar{Q}_0 \cdot Q_3 \cdot Q_1 \cdot (4096_2 + 4096_1 \cdot 1C) \cdot$$
$$\overline{Q_1 \cdot Q_2 \cdot Q_3} \cdot \overline{A(0)TESTER \cdot Q_1 \cdot Q_2 \cdot \bar{Q}_3}$$

Ecuațiile funcțiilor de ieșire sînt:

$$A(0)NC = A \cdot SPT = \bar{Q}_0 \cdot \bar{Q}_1 \cdot \bar{Q}_2 \cdot \bar{Q}_3 \cdot SPT$$

$$A(0)NM = B = \bar{Q}_0 \cdot Q_1 \cdot \bar{Q}_2 \cdot \bar{Q}_3$$

$$A(0)NR = C + H \cdot NC=NCF = Q_0 \cdot Q_1 \cdot \bar{Q}_2 \cdot \bar{Q}_3 + \bar{Q}_0 \cdot \bar{Q}_1 \cdot \bar{Q}_2 \cdot Q_3 \cdot NC=NCF$$

$$A(1)TESTER = D = Q_0 \cdot Q_1 \cdot Q_2 \cdot \bar{Q}_3$$

$$+1N_2 = D \cdot A(0)TESTER = Q_0 \cdot Q_1 \cdot Q_2 \cdot \bar{Q}_3 \cdot A(0)TESTER$$

$$+1NR = F \cdot \overline{NR=NRF} = Q_0 \cdot \bar{Q}_1 \cdot Q_2 \cdot \bar{Q}_3 \cdot \overline{NR=NRF}$$

$$+1NM = G \cdot \overline{NM=NMF} = Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3 \cdot \overline{NM=NMF}$$

$$+1NC = I = \bar{Q}_0 \cdot \bar{Q}_1 \cdot \bar{Q}_2 \cdot Q_3$$

$$A(0)SPT = H \cdot (NC=NCF) = \bar{Q}_0 \cdot \bar{Q}_1 \cdot \bar{Q}_2 \cdot Q_3 \cdot NC=NCF$$

Proiectarea UC-GAL

UC-GAL este unitatea de comandă ce preia conducerea tuturor operațiunilor pe parcursul desfășurării testului GALLOPING Vom face în continuare o prezentare succintă a acestui test.

Testul GALLOPING conține două etape: în prima etapă se consideră informația 11...1 ca dată de intrare (deci biestabilul D

ce dă datele de intrare va fi poziţionat pe "1"). In etapa a doua, datele de intrare sînt considerate oo...o (D="o").Pentru ambele etape, operaţiile executate sînt aceleaşi:

- se încercă în toate celulele datele (o)
- se citeşte celula o
- se complementează celula o
- se citeşte celula o apoi celula 1
- se citeşte celula o apoi celula 2
- .
- .
- .
- se citeşte celula o apoi celula cu adresa maximă
- se complementează celula o
- se citeşte celula 1 şi se complementează
- se citeşte celula 1 apoi celula 2
- se citeşte celula 1 apoi celula 3
- .
- .
- .
- se citeşte celula 1 apoi celula cu adresa maximă
- etc.(se complementează pe rînd toate celulele devenind bază a saltului)

Toate citirile sînt urmate de comparare, eventual generarea codului de eroare. Etapa a doua este identică, dar cu datele complementate.

Bistabilul asociat unităţii UC-GAL este bistabilul GAL. Prin poziţionarea lui pe "1" se apelează unitatea, care rămîne în buclă de aşteptare dacă bistabilul GAL e "o". Primele operaţiuni ale acestei unităţi sînt iniţializarea numărătorului de cicluri (A(o)NC) şi dirijarea ieşirilor bistabilului D (bistabil de date) spre intrarea de date a modulelor ((d)→id).Apoi bistabilul D se poziţionează pe "1" (pentru a se scrie informaţia iniţială "1" în toate celulele). Secţiunea de iniţializare continuă cu aducerea la "o" a numărătoarelor de module (A(o)NM) de rînduri (A(o)NR), de adresare curentă (A(o)N_2) şi de adresare în cadrul testului GALLOPING (A(o)N_1).

In cadrul acestui test adresarea se face consecutiv de la numărătoarele N_2 şi N_1; N_1 formează adresa celulei bază a saltului, iar N_2 a celulei ţintă trecîndu-se la intrarea de adresă

conținutul numărătorului N_2($(N_2)\longrightarrow$ia) și dîndu-se comanda de scriere, în celula o e memorată informația inițială ("1", conținutul bistabilului D). Această informație se înscrie și în următoarele celule, dîndu-se comanda de incrementare a numărătorului de adresare curentă ($+1N_2$) și comenda de scriere (SCR).

Cînd în toate celulele se află înscrisă informația inițială ($4o96_2$ e "1") ar urma conform mecanismului de testare să citim celula o. Pentru aceasta la intrarea de adresă este trecut conținutul numărătorului N_1 (acum "o"). Citirea, urmată de comparare și eventual de generarea codului de eroare, precum am văzut este condusă de unitatea UC-CODER; de aceea, unitatea UC-GAL apelează această unitate, prin poziționarea pe "1" a bistabilului asociat (A(1)TESTER). Cînd unitatea UC-CODER își termină activitatea, deci a fost citită celula o, această celulă va deveni baza șalturilor, de aceea, adresa ei, deci conținutul numărătorului N_1 trebuie trecut în numărătorul N_2 ($(N_1)\rightarrow N_2$), se complementează bistabilul D (CD) pentru a înscrie informația complementată în celula bază a șalturilor și se formează adresa celulei țintă prin incrementarea numărătorului N_2($+1N_2$); apoi se înscrie în celula bază a șaltului informația mai înainte pregătită (SCR). După declanșarea și terminarea ciclului respectiv (TC trece din "o" în "1" și din "1" în "o") controlul se cedează din nou unității UC-CODER, în vederea citirii celulei bază a șalturilor. Se verifică dacă adresa celulei bază, formată mai înainte este adresa maximă ($4o96_1$); dacă ultima celulă a memoriei nu a ajuns încă celula bază a șalturilor ($4o95_1$ e "o"), se trece la citirea celulei țintă. Pentru aceasta celula care adresează memoria trebuie să fie numărătorul celulelor țintă ($(n_2)\longrightarrow$ia). Datele sînt acum complementate (CD) pentru a fi pregătite în vederea formării următoarei celule bază. Se citește celula țintă prin apelarea unității UC-CODER(A(1)TESTER). După ce celula țintă e citită, se pregătește adresa noii celule țintă prin incrementarea numărătorului N_2($+1N_2$) și totodată se pregătește citirea din nou a celulei bază ($(n_1)\longrightarrow$ia). Verificăm acum dacă ultima adresă de celulă țintă formată este și ultima celulă a memoriei; dacă nu este, ($4o96_2$ e "o") se continuă șalturile cu aceiași bază pînă cînd se epuizează toate celule țintă ($4o96_2$ e "1"). Acum prin comanda SCR, se înscrie

în noua celulă bază a salturilor (a cărei adresă a fost pregătită mai sus) informaţia complementată. Pentru formarea adresei următoarei celule bază se incrementează numărătorul N_1(+1N_1). Se procedează la o nouă serie de salturi avînd ca celulă bază celula în care s-a înscris mai înainte informaţia complementată.

Lucrurile se repetă în continuare identic, pînă cînd ultima celulă a memoriei devine celulă bază a salturilor ($4o95_1$ e "1"), cînd se verifică dacă toate rîndurile de celule au fost supuse procedurii (NR=NRF). Dacă mai sînt rînduri netestate, mecanismul se repetă pînă la terminarea lor (NR= NRF e "1"). Acum se verifică dacă toate modulele au fost testate (NM=NMF). Dacă mai sînt module de testat, se trece la următorul modul netestat (+1NM). Cînd toate modulele au fost testate, dar a fost parcursă doar prima etapă (deci D este "o") se trece la parcurgerea celei de a doua etape. După terminarea ei (D="1"), se verifică dacă s-a parcurs numărul de cicluri dorit (fixat la cheile panoului); dacă nu, numărătorul de cicluri e incrementat şi toate operaţiile se reiau într-un nou ciclu. Cînd şi ultimul ciclu a fost terminat (NC=NCF e "1") unitatea de comandă UC-GAL intră în bucla de aşteptare iniţială, prin iniţializarea bistabilului asociat (A(o)GAL).

Organigrama de funcţionare este prezentată în figura 8.20. Diagrama stărilor curente şi a noilor stări sînt prezentate în figura 8.21 respectiv 8.22.

Ecuaţiile D sînt:

$$D_0=Q_1\cdot Q_4+Q_0\cdot\overline{Q}_2\cdot Q_3+Q_0\cdot\overline{Q}_1\cdot\overline{Q}_3+Q_0\cdot Q_2\cdot\overline{Q}_3+\overline{A(o)\text{TESTER}}\cdot Q_0\cdot Q_1\cdot Q_2+$$
$$4o96_2\cdot\overline{TC}\cdot\overline{Q}_1\cdot Q_2\cdot\overline{Q}_3\cdot\overline{Q}_4+\overline{\text{TESTER}}\cdot Q_1\cdot Q_2\cdot Q_3$$

$$=\overline{Q_1\cdot Q_4}\cdot\overline{Q_0\cdot\overline{Q}_2\cdot Q_3}\cdot\overline{Q_0\cdot\overline{Q}_1\cdot\overline{Q}_3}\cdot\overline{Q_0\cdot Q_2\cdot Q_3}\cdot\overline{\overline{A(o)\text{TESTER}}\cdot Q_0\cdot Q_1\cdot Q_2}\cdot$$
$$\overline{4o96_2\cdot\overline{TC}\cdot\overline{Q}_1\cdot Q_2\cdot\overline{Q}_3\cdot\overline{Q}_4}\cdot\overline{\overline{\text{TESTER}}\cdot Q_1\cdot\overline{Q}_2\cdot Q_3}$$

$$D_1=\overline{\overline{Q}_1\cdot Q_2}\cdot\overline{Q_0\cdot\overline{Q}_1\cdot Q_3}\cdot\overline{\overline{Q}_1\cdot\overline{Q}_4}\cdot\overline{\overline{Q}_0\cdot\overline{Q}_2\cdot\overline{Q}_3}\cdot\overline{4o96_2\cdot\overline{Q}_0\cdot Q_2\cdot Q_3}\cdot\overline{\overline{TC}\cdot Q_0\cdot Q_2\cdot\overline{Q}_3\cdot Q_4}\cdot$$
$$\overline{GAL\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot Q_4}\cdot\overline{\overline{DINC=NCF}\cdot Q_3\cdot Q_4}\cdot\overline{4o95_1\cdot NR=NRF\cdot NM=NMF\cdot Q_0\cdot\overline{Q}_2\cdot Q_3\cdot\overline{Q}_4}$$

$$D_2=Q_2\cdot Q_4+Q_1\cdot Q_2+Q_0\cdot Q_2+TC\cdot Q_2\cdot Q_3+\overline{4o96_2\cdot\overline{TC}}\cdot Q_2\cdot\overline{Q}_3+TC\cdot\overline{Q}_0\cdot Q_4+$$

$$\overline{4095_1}\cdot\overline{TC}\cdot Q_0\cdot Q_1\cdot Q_3\cdot\overline{Q}_4=\overline{Q_2\cdot Q_4\cdot\overline{Q}_1\cdot Q_2\cdot\overline{Q}_0\cdot\overline{Q}_2\cdot\overline{TC}\cdot Q_2\cdot Q_3\cdot\overline{4096_2}\cdot\overline{TC}\cdot Q_2\cdot\overline{Q}_3}$$

$$\overline{TC}\cdot\overline{Q}_0\cdot Q_4\cdot\overline{4095_1}\cdot\overline{TC}\cdot Q_0\cdot Q_1\cdot Q_3\cdot\overline{Q}_4$$

$$D_3=\overline{Q}_3\cdot\overline{Q}_4+Q_1\cdot\overline{Q}_3+\overline{Q}_0\cdot\overline{Q}_1\cdot\overline{Q}_3+Q_0\cdot\overline{Q}_4+\overline{D.NC=NCF}\cdot Q_1\cdot Q_4+\overline{TC}\cdot Q_2\cdot\overline{Q}_3+\overline{TC}\cdot\overline{4096_2}\cdot$$

$$\overline{Q}_0\cdot Q_1\cdot Q_2+\overline{4095_1}\cdot\overline{NR=NRF}\cdot Q_0\cdot\overline{Q}_2\cdot\overline{Q}_4=\overline{Q}_3\cdot\overline{Q}_4\cdot\overline{Q}_1\cdot\overline{Q}_3\cdot\overline{Q}_0\cdot\overline{Q}_1\cdot\overline{Q}_3\cdot\overline{Q}_0\cdot\overline{Q}_1\cdot\overline{Q}_4\cdot$$

$$\overline{D.NC=NCF}\cdot\overline{Q_1\cdot Q_4}\cdot\overline{TC}\cdot Q_2\cdot\overline{Q}_3\cdot\overline{\overline{TC}\cdot\overline{4096_2}\cdot\overline{Q}_0\cdot Q_1\cdot Q_2}\cdot\overline{4095_1\cdot\overline{NR=NRF}\cdot Q_0\cdot\overline{Q}_2\cdot\overline{Q}_4}$$

$$D_4=\overline{Q}_2\cdot\overline{Q}_3\cdot Q_4+\overline{Q}_0\cdot Q_1\cdot Q_4+Q_1\cdot\overline{Q}_2\cdot Q_4+\overline{Q}_0\cdot\overline{Q}_1\cdot\overline{Q}_2\cdot Q_3+Q_0\cdot\overline{Q}_1\cdot Q_2\cdot Q_4+\overline{4095_1}\cdot$$

$$NR=NRF\cdot NM=NMF\cdot Q_0\cdot Q_1\cdot\overline{Q}_2\cdot Q_3+\overline{TC}\cdot Q_2\cdot Q_3\cdot Q_4+A(o)TESTER\cdot Q_0\cdot\overline{Q}_1\cdot Q_2+TC\cdot$$

$$\overline{Q}_0\cdot Q_1\cdot Q_2\cdot\overline{Q}_3=\overline{Q}_2\cdot\overline{Q}_3\cdot Q_4\cdot\overline{Q}_0\cdot Q_1\cdot Q_4\cdot Q_1\cdot\overline{Q}_2\cdot Q_4\cdot\overline{Q}_0\cdot\overline{Q}_1\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot Q_0\cdot\overline{Q}_1\cdot Q_2\cdot Q_4\cdot$$

$$4095_1\cdot NR=NRF\cdot NM=NMF\cdot Q_0\cdot Q_1\cdot\overline{Q}_2\cdot Q_3\cdot\overline{TC}\cdot Q_2\cdot Q_3\cdot Q_4\cdot A(o)TESTER\cdot$$

$$Q_0\cdot\overline{Q}_1\cdot Q_2\cdot TC\cdot\overline{Q}_0\cdot Q_1\cdot Q_2\cdot\overline{Q}_3$$

Ecuațiile comenzilor sînt:

$$A(o)NC=\overline{(d)\longrightarrow id}=A.UAL=Q_0\cdot Q_1\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot Q_4\cdot UAL$$

$$A(1)D=B=Q_0\cdot\overline{Q}_1\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot Q_4$$

$$A(o)NM=C=Q_0\cdot\overline{Q}_1\cdot\overline{Q}_2\cdot Q_3\cdot Q_4$$

$$A(o)NR=X=Q_0\cdot\overline{Q}_1\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot\overline{Q}_4$$

$$A(o)N_1=A(o)N_2=E=Q_0\cdot Q_1\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot\overline{Q}_4$$

$$\overline{(n_2)\longrightarrow is}=F=Q.\overline{4095_1}=\overline{F.Q.4095_1}=\overline{F}$$

$$SCR=G+M+U=\overline{\overline{G}.\overline{M}.\overline{U}}$$

$$+1N_2=I+L+S.A(o)TESTER=\overline{\overline{I}.\overline{L}.\overline{SA(o)TESTER}}$$

$$\overline{(n_1)\longrightarrow is}=K+S.A(o)TESTER=\overline{\overline{K}.\overline{S.A(o)TESTER}}$$

$$A(1)TESTER=K+P+S=\overline{\overline{K}.\overline{P}.\overline{S}}$$

$(N_1)\longrightarrow N_2$=C.A(o) TESTER

CD=L+Q.$\overline{4o95_1}$.$\overline{TC}$+Q.$4o95_1$.$\overline{NR=NRF}$+Q.$4o95_1$.NR=NRF.$\overline{NM=NMF}$+S.A(o)

TESTER.T.$4o96_2$=I.Q.$\overline{4o95_1}$.$\overline{TC}$.Q.$4o95_1$.$\overline{NR=NRF}$.Q.$4o95_1$.NR=NRF

$\overline{\overline{NM=NMF}.\overline{S.A(o)TESTER}.\overline{T.4o96_2}}$.

+1NR=Q.$4o95_1$.$\overline{NR=NRF}$

+1NM=Q.$4o95_1$.NR=NRF.$\overline{NM=NMF}$

+1NC=R.D^*.$\overline{NC=NCF}$

A(o)GAL=R.D^*.NC=NCF

+1N_1=V

Proiectarea UC

UC este unitatea de comandă care dă succesiunea tuturor solicitărilor sau testelor la care urmează a fi supuse modulele de memorie. Ea apelează unităţile: subprogram de îmbătrînire SPI; subprogram de testare SPT; unitatea de comandă GALLO-PING, UC-GAL. Doar unitatea de generare a codurilor de eroare UC-CODER nu poate fi apelată direct de către unitatea de comandă UC.

UC îşi începe activitatea după ce bistabilul START este poziţionat pe "1" de către un semnal venit de la microprocesor Cît timp acest semnal nu soseşte, bistabilul START rămîne poziţionat pe "o", iar unitatea de comandă stă în bucle de aşteptare.

După ce START e poziţionat pe "1", UC iniţializează numărătorul de adresare curentă (A(o)N_2) şi numărătorul de teste (A(o)Z); totodată se pregăteşte adresarea de la numărătorul de adresare curentă ($(n_2)\longrightarrow$ia) şi intrarea datelor de la cheile de date ale panoului ((gad)$\longrightarrow$id). Se verifică în continuare dacă s-a cerut o îmbătrînire la scriere sau la citire (bistabilul I-CIT este poziţionat pe "o" sau pe "1" de către comutator cu trei poziţii de pe panou). Dacă I-CIT este pe "1", bistabilul de citire/scriere este poziţionat pe "1" (A(1)BC/S). In acest moment UC apelează unitatea ce conduce activitatea de îmbătrînire, prin poziţionarea pe "1" a bistabilului asociat (A(1)SPI). După ce unitatea SPI îşi încheie

activitatea, este testat bistabilul TEST, poziţionat pe "1" de către un semnal venind de la microprocesor.

Dacă acest bistabil este pe "o", se apelează din nou unitatea SPI pentru un nou ciclu de îmbătrînire. Cînd TEST e "1", se incrementează cu 1 numărătorul de teste (+1Z) conţinutul său devenind acum 1. După cum am mai arătat, numărătorul Z conţine:

- ooo la îmbătrînire
- ool la scriere,citire,comparare cu conţinutul cheilor de date
- olo la scriere,citire, comparare cu "o"
- oll la scriere,citire, comparare cu "1"
- loo la scriere,citire, comparare cu adresa curentă
- lol la testul GALLOPING

In acest moment deci ar urma memorarea conţinutului cheilor de date, citirea informaţiei memorate, compararea cu conţinutul cheilor şi eventual genererea codului de eroare. Pentru aceasta, bistabilul de citire/scriere este iniţializat (A(o)BC/S) indicînd că urmează o scriere. Inscrierea conţinutului cheilor de date se face printr-un ciclu de îmbătrînire la scriere, deci se apelează unitatea SPI(A(1)SPI). După ce aceasta îşi încheie activitatea, deci a fost memorat conţinutul cheilor de date ar urma comparareş şi generarea codului de eroare. Pentru executarea acestora este apelată unitatea specializată în aceste activităţi (A(1)SPT). Se verifică (după ce SPI predă controlul (A(o)SPT)), dacă conţinutul numărătorului de teste este 1; acesta fiind într-adevăr 1, se pregăteşte percurgerea următorului test, corespunzător conţinutului numărătorului de teste după proxima incrementare, deci scriere, citire, comparare cu "o"; pentru aceasta se generează comenzile A(o)D şi (d)⟶id; numărătorul de teste e incrementat, conţinutul său fiind acum 2, deci lucrurile se repetă cu datele complementate. La terminarea şi a acestui test, se verifică dacă conţinutul numărătorului de teste este 1; bineînţeles nu este (Z_1="o") apoi se verifică dacă este 2. Dacă da, (Z_2="1") se complementează bistabilul de date, deoarece urmează testul de înscrâere, citire, comparare cu "1". După terminarea acestuia, conţinutul numărătorului de teste va fi 3 (Z_3="1") deci urmează testul de scriere, citire, comparare

cu adresa curentă. (La care conținutul numărătorului va fi 4) pentru aceasta la intrarea de date se îndrumă adresa curentă ((ia)→id) și lucrurile se reiau. După terminarea acestor date (deci Z_1 este "o", Z_2="o", Z_3="o"). Se apelează unitatea de comandă UC-GAL, pentru desfășurarea testului GALLOPING (A(1)GAL). După ce aceasta își termină activitatea,(GAL="o"), unitatea de comandă intră în buclă de așteptare inițială.

Urmează să tratăm cazul când, de la panou, nu se cere îmbătrînire la citire (I-CIT="o"). In acest caz, se inițializează bistabilul citire/scriere BC/S se verifică dacă se cere îmbătrînire la scriere; dacă da (I-SCR este "1") se reia totul de la prima apelare a unității SPI, pentru îmbătrînirea la scriere. Dacă nu se cere nici îmbătrînire la scriere (I-SCR este "o"), prin poziționarea pe "1" a bistabilului de îmbătrînire regenerare (A(1)BI-REG) se procedează la un regim de regenerare, ce durează până când bistabilul TEST e poziționat pe "1" de către un semnal de la microprocesor, după care bistabilul de îmbătrînire regenerare este inițializat (A(o)BI-REG). In continuare celelalte teste se desfășoară după cum au fost descrise mai sus.

In cazul regimului de îmbătrînire la scriere sau citire, durata acestuia poate varia, după necesități, lucru realizat prin bistabilul TEST care poate fi poziționat pe "1" la un moment sau altul de către microprocesor, determinînd începerea celorlalte teste.

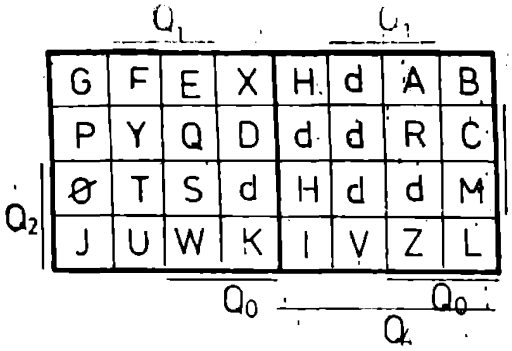
Organigrama de funcționare este prezentată în fig.8.23, iar diagramele pentru stările curente și următoare în fig. 8.24 respectiv 8.25.

In continuare sînt date ecuațiile de intrare și ecuațiile funcțiilor de comandă:

$$\bar{\vartheta}_0=Q_0\cdot Q_3+\bar{Z}_1\cdot\bar{Z}_2\cdot\bar{Z}_3\cdot Q_0\cdot Q_2+GAL\cdot Q_0\cdot\bar{Q}_2+\overline{SPI}\cdot\overline{V_{MOS}}\cdot\bar{Q}_1\cdot Q_3$$

$$D_0=\overline{\overline{Q_0\cdot Q_3}\cdot\bar{Z}_1\cdot\bar{Z}_2\cdot\bar{Z}_3\cdot Q_0\cdot Q_2\cdot GAL\cdot Q_0\cdot\bar{Q}_2\cdot\overline{\overline{SPI}\cdot\overline{V_{MOS}}\cdot\bar{Q}_1\cdot Q_3}}$$

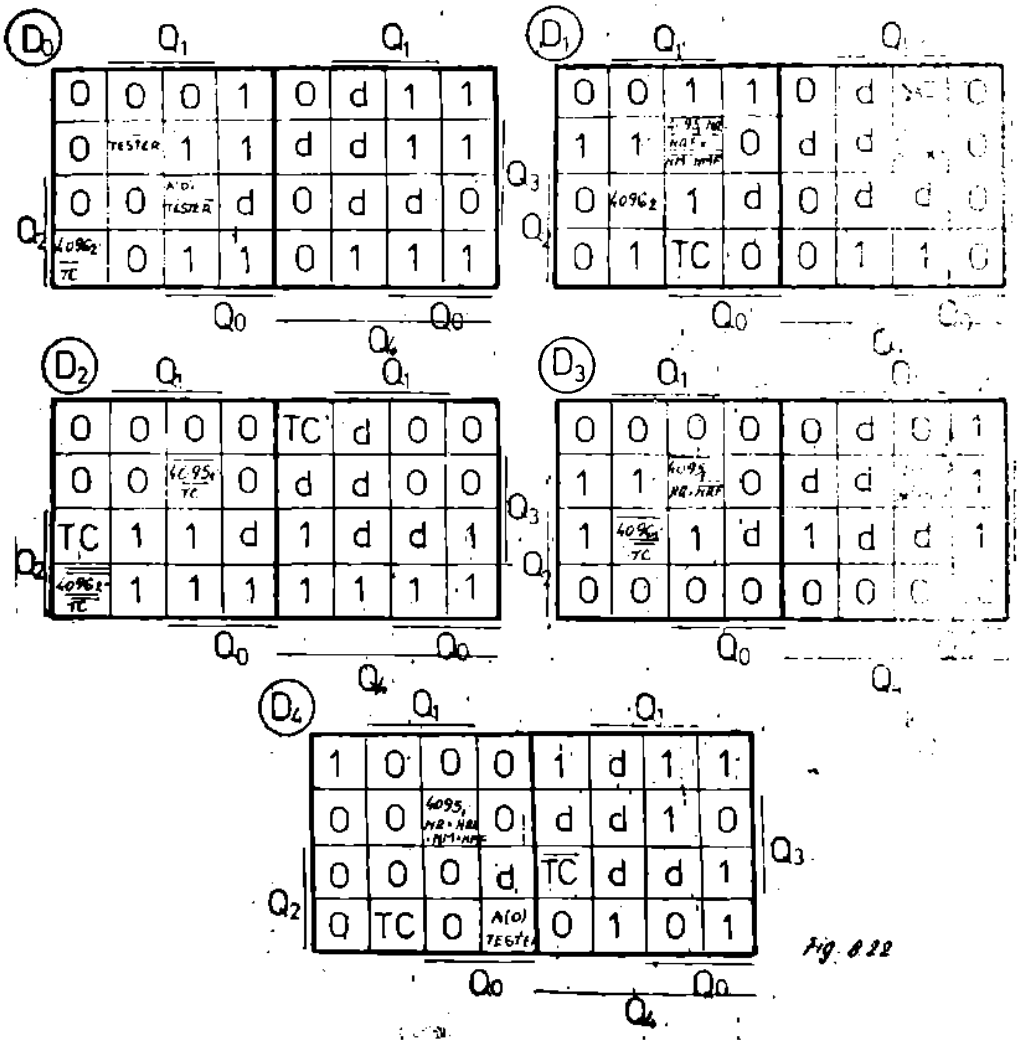
$$D_1=\overline{START\cdot\overline{I\text{-}CIT}\cdot\overline{I\text{-}SCR}\cdot\bar{Q}_1\cdot\bar{Q}_2+\overline{GAL}\cdot Q_0\cdot\bar{Q}_2+\overline{SPI}\cdot Q_3\cdot\bar{Q}_1+Q_3\cdot Q_1\cdot\bar{Q}_0+}$$
$$+\overline{\overline{SPI}\cdot\overline{TEST}\cdot\bar{Q}_1\cdot\bar{Q}_3\cdot Q_2}$$



Q_2 \ Q_1Q_0	00	01	11	10
11	G	F	E	X
10	P	Y	Q	D
01	Ø	T	S	d
00	J	U	W	K

Q_2 \ Q_1Q_0	00	01	11	10
11	H	d	A	B
10	d	d	R	C
01	H	d	d	M
00	I	V	Z	L

Fig. 8.21.



D_0

Q_2 \ Q_1Q_0	00	01	11	10
11	0	0	0	1
10	0	TESTER	1	1
01	0	0	A/D TESTER	d
00	4096 $\overline{TC}$	0	1	1

Q_2 \ Q_1Q_0	00	01	11	10
11	0	d	1	1
10	d	d	1	1
01	0	d	d	0
00	0	1	1	1

D_1

Q_2 \ Q_1Q_0	00	01	11	10
11	0	0	1	1
10	1	1	4095 NR·HR	0
01	0	4096	1	d
00	0	1	TC	0

Q_2 \ Q_1Q_0	00	01	11	10
11	0	d	[illegible]	0
10	0	d	d	0
01	0	d	d	0
00	0	1	1	0

D_2

Q_2 \ Q_1Q_0	00	01	11	10
11	0	0	0	0
10	0	0	4095 $\overline{TC}$	0
01	TC	1	1	d
00	4096 $\overline{TC}$	1	1	1

Q_2 \ Q_1Q_0	00	01	11	10
11	TC	d	0	0
10	d	d	0	0
01	1	d	d	1
00	1	1	1	1

D_3

Q_2 \ Q_1Q_0	00	01	11	10
11	0	0	0	0
10	1	1	4095 NR·HR	0
01	1	4096 $\overline{TC}$	1	d
00	0	0	0	0

Q_2 \ Q_1Q_0	00	01	11	10
11	0	d	[illegible]	1
10	d	d	[illegible]	1
01	1	d	d	1
00	0	0	0	[illegible]

D_4

Q_2 \ Q_1Q_0	00	01	11	10
11	1	0	0	0
10	0	0	4095 NR·HR·HF	0
01	0	0	0	d
00	0	TC	0	A(0) TESTER

Q_2 \ Q_1Q_0	00	01	11	10
11	1	d	1	1
10	d	d	1	0
01	$\overline{TC}$	d	d	1
00	0	1	0	1

Fig. 8.22

$$D_1=START.\overline{I\text{-}CIT}.\overline{I\text{-}SCR}.\bar{Q}_1.\bar{Q}_2.\overline{GAL}.Q_0.\bar{Q}_2.\overline{SPI}.\bar{Q}_1.Q_3.\bar{Q}_0.Q_1.Q_3.\overline{SPI.TEST}.\bar{Q}_1.Q_2.\bar{Q}_3$$

$$D_2=Q_3+Q_2.Q_0+TEST.\bar{Q}_0.Q_1+START(\overline{\overline{I\text{-}CIT}.\overline{I\text{-}CIT}.I\text{-}SCR}).Q_0.Q_1+\bar{Z}_1.\bar{Z}_2.\bar{Z}_3.Q_2$$

$$D_2=\bar{Q}_3.\bar{Q}_0.Q_3.\overline{TEST}.\bar{Q}_0.Q_1.\overline{START.\overline{I\text{-}CIT}.\overline{I\text{-}SCR}.\bar{Q}_0.\bar{Q}_1.\bar{Z}_1.\bar{Z}_2.\bar{Z}_3}.Q_2$$

$$D_3=Q_3.\bar{Q}_0+Q_1.Q_2.\bar{Q}_0+\overline{A(o)SPI}.Q_3$$

$$D_3=Q_3.\bar{Q}_0.Q_1.Q_2.\bar{Q}_2.\overline{A(o)SPT}.Q_3$$

$(n_2)=ia=(gad)\longrightarrow id=A(o)N_2=A.START=\bar{Q}_0.\bar{Q}_1.\bar{Q}_2.\bar{Q}_3.START$

$(o)BC/S=A.START.\overline{I\text{-}CIT}+1=\overline{\bar{Q}_0.\bar{Q}_1.\bar{Q}_2.\bar{Q}_3.START.\overline{I\text{-}CIT}}+\bar{Q}_0.Q_1.Q_2.Q_3.=$

$\bar{Q}_0.\bar{Q}_1.\bar{Q}_2.\bar{Q}_3.START.\overline{I\text{-}CIT.\bar{Q}_0.Q_1.Q_2.Q_3}$

$(1)BC/S=A.START.I\text{-}CIT=\bar{Q}_0.\bar{Q}_1.\bar{Q}_2.START.I\text{-}CIT$

$(1)BI\text{-}REG=A.START.\overline{I\text{-}CIT}.\overline{I\text{-}SCR}=\bar{Q}_0.\bar{Q}_1.\bar{Q}_2.\bar{Q}_3.START.\overline{I\text{-}CIT}.\overline{I\text{-}SCR}$

$(1)SPI=C+I=\bar{Q}_0.\bar{Q}_1.Q_2.\bar{Q}_3+\bar{Q}_0.Q_1.Q_2.Q_3=\overline{\bar{Q}_0.\bar{Q}_1.Q_2.\bar{Q}_3.\bar{Q}_0.Q_1.Q_2.Q_3}$

$(o)BI\text{-}REG=B.TEST=\bar{Q}_0.Q_1.\bar{Q}_2.\bar{Q}_3.TEST$

$lZ=D+G.\bar{Z}_1.\bar{Z}_2.\bar{Z}_3=\overline{\bar{D}.G.\bar{Z}_1.\bar{Z}_2.\bar{Z}_3}=\overline{\bar{Q}_0.Q_1.Q_2.\bar{Q}_3.Q_0.Q_1.Q_2.\bar{Q}_3.\bar{Z}_1.\bar{Z}_2.\bar{Z}_3}$

$(1)SPT=F=Q_0.Q_1.Q_2.Q_3$

$(o)D=(d)\longrightarrow id=G.Z_1=Q_0.Q_1.Q_2.\bar{Q}_3.Z_1$

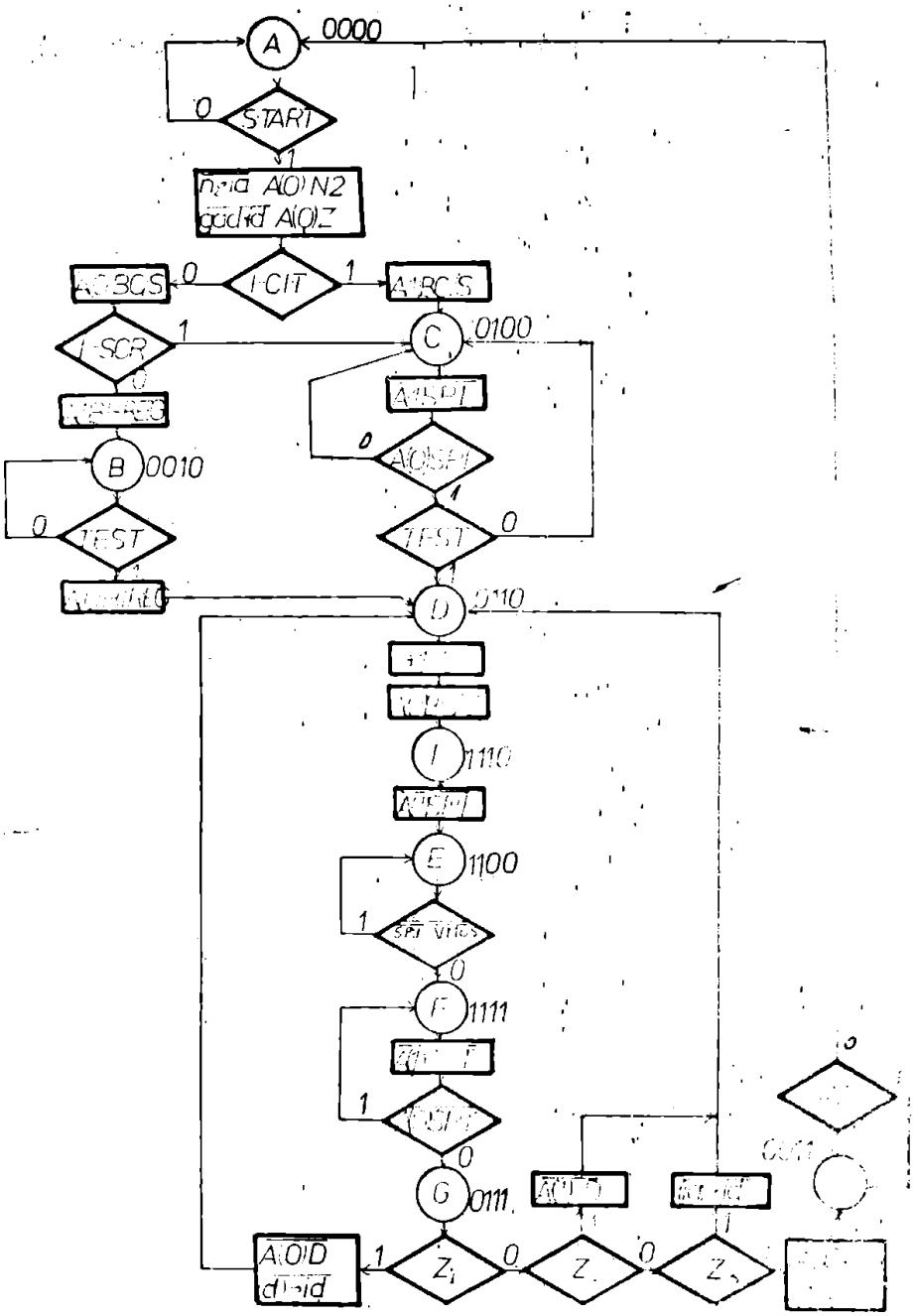


Fig. 8.23.

$A(1)D=G.\overline{Z}_1.Z_2=Q_o.Q_1.Q_2.\overline{Q}_3.\overline{Z}_1.Z_2$

$ia) \rightarrow id=G.\overline{Z}_1.\overline{Z}_2.Z_3=Q_o.Q_1.Q_2.\overline{Q}_3.\overline{Z}_1.\overline{Z}_2.Z_3$

$A(1)GAL=G.\overline{Z}_1.\overline{Z}_2.\overline{Z}_3=Q_o.Q_1.Q_2.\overline{Q}_3.\overline{Z}_1.\overline{Z}_2.\overline{Z}_3$

OBS. Referitoare la sintetizarea schemelor functiilor de iesire.

Existînd semnale care constituie funcţii de ieşire la mai multe unităţi de comandă, se impune scrierea ecuaţiilor unificate ale acestor funcţii. In acest scop se vor utiliza indici a căror semnificaţie se dă în continuare:

i - SPI
t - SPI
c - UC
e - UC-CODER
g - UC GALLOPING

$SCR=SCR_i+SCR_g$

$CIT=CIT_i+CIT_e$

$+1N_2=(+1N_{2i})+(+1N_{2t})+(1N_{2g})$

$+1R=(+1NR_i)+(+1NR_t)+(+1NR_g)$

$+1NM=(+1NM_t)+(+1NM_g)$

$+1NC=(+1NC_t)+(+1NC_g)$

$A(o)NR=A(o)NR_i+A(o)NR_t+A(o)NR_g$

$A(o)NC=A(o)NC_t+A(o)NC_g$

$A(o)NM=A(o)NM_t+A(o)NM_g$

$A(1)TESTER=A(1)TESTER_t+A(1)TESTER_g$

$(n_2) \rightarrow ia=(n_2) \longrightarrow ia_c+(n_2) \longrightarrow ia_g$

$(d) \rightarrow id=(d) \rightarrow id_c+(d) \rightarrow id_g$

$A(1)D=A(1)D_c+A(1)D_g$

$A(o)N_2=A(o)N_{2c}+A(o)N_{2g}$

Aceste ecuaţii sînt doar principiale, fără a ţine cont de polarităţile semnalelor. Transcriem în continuare aceste ecuaţii sub forma din care se pot sintetiza:

$$SCR=\overline{Q_{0i}\cdot Q_{1i}\cdot Q_{2i}}\cdot SPI\cdot BC/S\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}$$

$$+1NR=\overline{Q_{0i}\cdot Q_{1i}\cdot Q_{2i}}\cdot NR=NRF\cdot Q_{0t}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot NR=NRF\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}\cdot 4095_1\cdot NR=NRF$$

$$A(o)N_2=\overline{Q_{0c}\cdot Q_{1c}\cdot Q_{2c}\cdot Q_{3c}}\cdot START\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}$$

$$+1NM=\overline{Q_{2t}\cdot Q_{3t}}\cdot NM=NMF\cdot Q_{0t}\cdot Q_{1t}\cdot 4095_1\cdot NR=NRF\cdot NM=NMF\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{4g}\cdot Q_{3g}$$

$$CIT=\overline{Q_{0i}\cdot Q_{1i}\cdot Q_{2i}}\cdot SPT\cdot BC/S\cdot Q_e\cdot TESTER$$

$$A(o)NR=\overline{Q_{0i}\cdot Q_{1i}\cdot Q_{2i}}\cdot NR=NRF\cdot Q_{0t}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot Q_{0t}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot NC=NCF\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}$$

$$A(1)D=\overline{Q_{0c}\cdot Q_{1c}\cdot Q_{2c}\cdot Q_{3c}\cdot Z_1\cdot Z_2}\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}$$

$$1NC=\overline{Q_{0t}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}\cdot D}\cdot NC=NCF$$

$$(n_2)\longrightarrow ia=\overline{Q_{0c}\cdot Q_{1c}\cdot Q_{2c}\cdot Q_{3c}}\cdot START\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{4g}\cdot Q_{3g}\cdot Q_{4g}\cdot 4095_1\cdot Q_{0g}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}$$

$$(d)\longrightarrow id=\overline{Q_{0c}\cdot Q_{1c}\cdot Q_{2c}\cdot Q_{3c}}\cdot Q_1\cdot A(o)NC$$

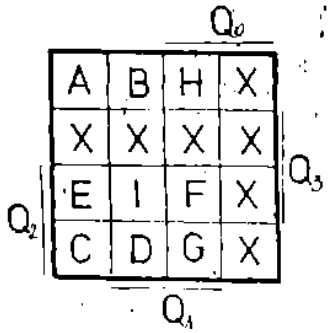


Fig. 8.24.

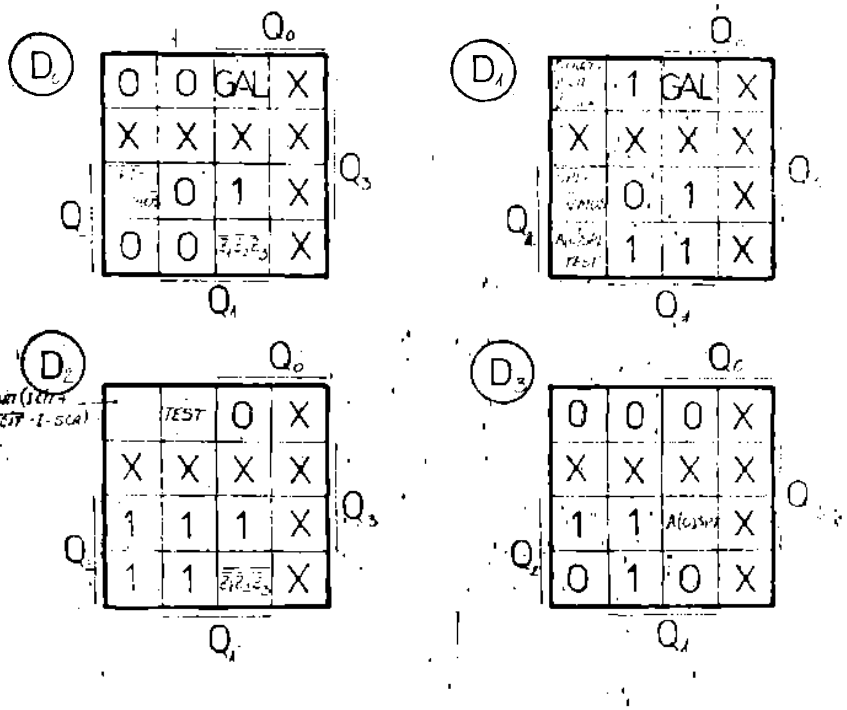


Fig. 8.25

$$+1N_2=\overline{Q_{ot}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot TESTER}\cdot\overline{C_{oi}\cdot Q_{1i}\cdot Q_{2i}\cdot TESTER}\cdot\overline{Q_{og}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}}\cdot\overline{Q_{og}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}}$$

$$A(o)NC=\overline{Q_{ot}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot SFT}\cdot\overline{Q_{og}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}\cdot GAL}$$

$$A(o)NM=\overline{Q_{ot}\cdot Q_{1t}\cdot Q_{2t}\cdot Q_{3t}\cdot Q_{og}\cdot Q_{1g}\cdot Q_{2g}\cdot Q_{3g}\cdot Q_{4g}}$$

8.2. Structură de comandă cu execuţii paralele cu mai *multe* DC "slave" supervizate de două DC "master" şi ierarhizare variabilă (flexibilă), utilizată la achiziţie/distribuţie de date în timp real.

8.2.1. Introducere

Sistemul de achiziţie/distribuţie supervizat şi comandat de structura de comandă din titlu, are funcţiuni multiple şi anume:

- achiziţia de parametri analogici (date analogice)
- achiziţia de parametri numerici (date numerice)
- generarea de parametri numerici interni funcţie de parametrii de intrare
- efectuarea de corecţii de scară în două etape:
  - corecţie analogică
  - corecţia numerică
- efectuarea conversiei din binar în cod BCD în vederea afişării zecimale a parametrilor, pentru control vizual
- efectuarea unei noi conversii din binar paralel în binar serie (pentru transmisie sau stocare)
- gruparea mai multor seturi de parametri sub formă de blocuri şi descărcarea acestor blocuri pe un suport de informaţie magnetic (bandă, casetă, disc, ş.a).
- redarea (citirea) blocurilor înregistrate anterior
- despachetarea blocurilor
- reconvertirea serie-paralel

- efectuarea conversiei numeric-analogice (numai la parametrii de tip analogic)
- distribuirea parametrilor analogici (succesiuni de eşantioane memorate analogic şi apoi filtrate)şi a parametrilor numerici
- utilizarea parametrilor distribuiţi la comanda unui proces analog-numeric (hibrid) cu program preînregistrat.
- afişarea numerică zecimală a parametrilor distribuiţi (în unităţi de mărimi fizice) şi afişarea analogică (trasator de curbe, osciloscop cu remanenţă) pentru controlul vizual sau pentru obţinerea unui document de istorie a procesului.

Sistemul realizat practic are o structură originală şi experimentat pentru înregitrarea/reproducerea parametrilor sudării automate[anexa 2, FOTO 2]poate fi utilizat şi în alte aplicaţii.

Rolul acestui sistem este de a genera un document al "istoriei" unei suduri atît pentru a certifica calitatea ei cît şi pentru a servi la reproducerea în mod automat a unor suduri. Sistemul este destinat a fi utilizat la fabricarea automată a unor componente de instalaţii din industria chimică şi din cadrul centralelor nuclearo-electrice.

Parametrii esenţiali ai unei suduri sînt:

- curentul de sudare
- tensiunea arcului electric
- viteza de avans a tractorului de sudare, ca funcţii de timp.

Sistemul care va fi descris în continuare poate înregistra 7 parametri cu semn (3 curenţi, 2 tensiuni, 2 viteze) precum şi timpul, ca parametrul generat intern, în acest sistem automat.

Parametrii se prezintă sub formă de tensiuni analogice unificate, cu semn, cuprinse între 0 - 10 V.

Aceştia sînt convertiţi în formă numerică, afişaţi zecimal şi memoraţi binar pe o unitate de casetă magnetică UCM 101-2.

De pe casetă, se convertesc din nou în formă analogică, pentru a putea comanda o instalaţie de sudură automată, pentru a fi vizualizaţi în formă grafică (la un trasator de curbe), sau se reconvertesc din nou în formă numerică pentru a fi afişaţi în zecimal.

In continuare, după o prezentare succintă a întregii instalaţii, nu se va prezenta în detaliu decît partea mai interesantă, şi anume unitatea de comandă.

8.2.2. Schema bloc a sistemului. Descrierea funcţionării

In fig.8.26 este prezentată schema bloc a IR-BM (sistem de înregistrare-redare utilizînd bandă magnetică).

Semnificaţiile blocurilor sînt următoarele:

1 - multiplexor care selectează parametrii sudării în curs sau parametrii preînregistraţi numeric şi redaţi analogic

2 - multiplexor analogic pe post de multiplexor numeric pentru semnalele parametrilor.

3 - divizor pentru corecţia analogică de scară şi multiplexor analogic al parametrilor de intrare.

4 - bloc de urmărire-eşantionare-memorare

5 - convertor analog-numeric pe 6 cifre binare.

6 - numărător pe post de "contor de timp" care avansează cu o unitate la fiecare set de parametri selecţaţi.

7 - multiplexor care selectează un parametru de sudare (cel deja multiplexat în blocul 3), sau timpul.

8 - bistabil care generează date în completarea parametrilor unei suduri care se termină "0" - pe post de date; "1" -pe post de marcă - multiplexate de 9 ori cît este cuvîntul serie, memorat pe casetă).

9 - un nou nivel de multiplexare - care permite intrarea datelor generate de bistabilul 8 spre registrul tampon 10.

10 - registru tampon care asamblează cuvinte de nouă biţi, dacă vin serie de la casetă sau le dezasamblează pentru memorare pe casetă, dacă vin paralel de la convertorul A/N.

11 - registru de deplasare stînga-dreapta care serveşte la efectuarea corecţiei numerice de scară, pentru afişarea în unităţi de mărimi fizice reale.

12 - numărător binar invers

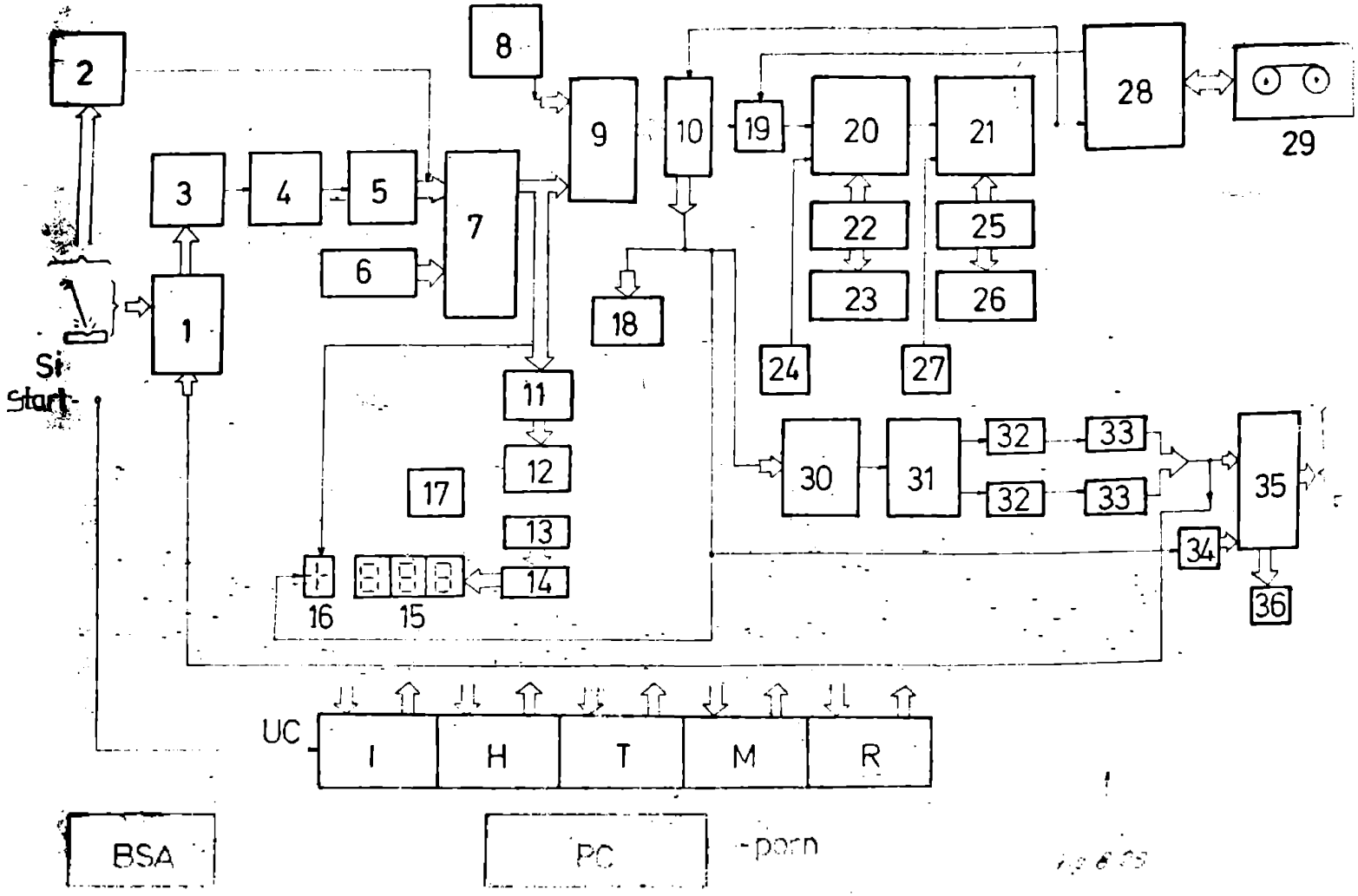
13 - numărător BCD direct

14 - decodificator pe 3 ranguri zecimale BCD-7 segmente

15 - afişaj zecimal pe 3 ranguri

16 - afişaj semn.

17 - bistabil de autorizare a numărării în numărătoarele 12 şi 13 pentru conversia binar-BCD.


2
8
9
10
19
20
21
28
29
3
4
5
7
22
25
6
23
26
18
1
Si
start
24
27
11
12
17
30
31
32
33
35
13
34
14
16
15
36
UC
I
H
T
M
R
BSA
PC
-porn

18 - detector de mărci (11...11)

19 - multiplexor date-serie (din MT-bloc 10 sau de la interfaţa casetei - bloc 28).

20 - memorie RAM conectată în timp real la parametrii de sudare.

21 - memorie RAM ce serveşte la descărcarea datelor sub formă de blocuri, pe casetă.

22,25 - registre de adresă

23,26 - detectoare pentru anumite adrese (704^{1}_{10}, 720^{1}_{10}, 720^{2}_{10}).

24,27 - baza de timp (DCLm) pentru $RAM_{1,2}$.

28 - interfaţa sistemului cu unitatea de casetă magnetică

29 - unitatea de casetă magnetică UCM 101-2.

30 - convertor numeric/analogic

31 - demultiplexor analogic pentru parametrii redaţi.

32 - blocuri de eşantionare-memorare.

33 - blocuri de filtrare

34 - demultiplexor pentru semnele parametrilor redaţi.

35 - sistem de burse de sudare şi tractor pentru sudare automată.

36 - dispozitiv de trasare grafică

S_i - instalaţie de sudare de la care se generează parametrii ce urmează a fi înregistraţi

S_e - instalaţie de sudare automată după program preînregistrat.

UC - unitate de comandă numerică a întregului sistem, din care:

- I - dispozitiv de comandă (dc) al înregistrării
- H - d.c. al corecţiei numerice de scară, conversiei binar - BCD şi afişării zecimale.
- T - d.c. al încărcării serie-paralel al MT (bloc 10), a descărcării sale serie-paralel şi a deplasărilor stînga-dreapta.
- A - d.c. al celor două memorii RAM cuplate între ele şi al transferurilor de date aferente
- R - d.c. al redării
- PC - panou de comandă şi vizualizare
- BSA- blocul sistemului de alimentare.

Funcţionarea de ansamblu a sistemului este următoarea: la apariţia tranziţiei 0→1 a semnalului "start", blocul I, gene-

rează semnale de selecţie succesivă, a parametrilor şi semnelor lor spre multiplexoarele 1 şi 2, apoi spre 3. Apoi dă comanda de eşantionare - memorare spre blocul 4 şi declanşează conversia analog-numerică la blocul 5. După răspunsul blocului 5 (sfîrşit conversie), multiplexorul 7vselectează parametrul curent, sau timpul, generat numeric în blocul 6, pentru a merge mai departe la sistemul de vizualizare sau spre lanţul care conduce în final la înregistrarea pe casetă.

Intrucît în blocul 3 s-a făcut corecţia analogică de scară (prin divizare), blocul 11 (registrul de deplasare) va efectua corecţie numerică de scară prin divizare sau multiplicare cu 2.

Conversia binar - BCD este asigurată printr-o numărare simultană, directă în numărătorul 13, inversă în numărătorul 12, pînă la anularea conţinutului acestuia.

Blocul 12 este încărcat paralel din registrul 11.

Bistabilul 17 autorizează aceste două *numărări* pînă la încheierea conversiei.

Blocul 14 converteşte ieşirile BCD ale blocului 13 şi le trimite spre afişajul zecimal pe 7 segmente, 15.

Afişajul 16, permite vizualizarea semnului parametrului afişat.

Cuvîntul binar-paralel de 6 cifre plus semn, trece prin blocurile 9 şi 10, iar apoi în serie, bit cu bit prin blocurile 19, 20,21,26,29.

Dacă are loc o tranziţie 1$\longrightarrow$0 a semnalului "start",atunci datele trec prin fenomene tranzitorii (datorită opririi procesului de sudare), şi valorile acestora sînt nesemnificative. Din acest moment, bistabilul b se poziţionează pe "0", multiplexorul 9, selectează intrările conectate la acest bistabil (de 9 ori), adică calea de sus în fig.8,6pentru a completa cuvinte de 9 biţi în RT (10), $RAM_{1,2}$(20,21). Ultimul cuvînt de 9 biţi care se mai memorează este 11 ... 11, el constituind marca de sfîrşit-sudură. Acest lucru se realizează prin poziţionarea pe "1" a bistabilului b.

Cuvintele întrare paralel în blocul 10 (RT) sînt trimise serie prin multiplexorul de 1 bit, 19 pe calea din stînga, spre RAM_1(20). Cînd RA_1(22) ajunge la adresa considerată de noi finală (720), acest lucru este sesizat de blocul 23, care declanşează pornirea dispozitivului de comandă k, care va conduce şi controla transferul datelor din RAM_1 în RAM_2 într-un timp scurt, mai

scurt decît intervalul între două selecţii a parametrilor de intrare. Cînd RAM_2 este plin, adică RA_2 (25) conţine adresa 720, fapt sesizat de circuitul 26, se aduce la "0 ... 0" RA_1, pentru a putea permite în continuare, încărcarea RAM_1 cu cuvinte serie de 9 biţi reprezentînd parametri de sudare. Tot în acest moment se declanşează circuitele din interfaţa 28, care citeşte bit cu bit informaţie din RAM_2 şi o memorează sub formă de bloc, pe casetă, oprind apoi caseta.

Blocurile DCLM 24 şi 27, precum şi $RA_{1,2}$ (22,25) pot fi comandate atît de blocul M cît şi de blocul 28.

Deci pînă acum s-au descris fenomenele care au loc la înregistrare.

La redare, declanşată de la panpul PC, prin comutatorul RcDARe, se declanşează blocul R care trimite o comandă de pornire a casetei, la blocul 28. Datele citite de pe casetă trec pe calea 19,20,21,10, de o manieră similară ca la înregistrare Din RT (blocul 10), cuvîntul paralel de 9 biţi este trimis astfel: cei 8 biti reprezentînd mărimea, sînt trimişi spre detectorul de mărci 18 care la apariţia cuvîntului 11 ... 11, acţionează asupra blocului R, oprind redarea. Bitul semn este trimis spre demultiplexorul 34, iar de aici la instalaţia de sudură automată (35 şi S_e). Tot bitul semn mai este trimis şi la sistemul de afişaj al semnului 16 (care mai conţine de fapt şi nişte circuite de selecţie, nefigurate).

Cei 8 biţi reprezentînd mărimea numărului sînt trimişi şi la blocul 30 (convertor N/A) iar de aici la demultiplexorul 31 şi apoi pentru fiecare ieşire prin cîte un bloc 32 şi 33 unde se face eşantionarea şi memorarea respectiv filtrarea şi refacerea nivelelor (refacerea corecţiei analogice de scară). In final şi aceste semnale ajung la blocurile 35 şi S_e. Dacă se doreşte o trasare grafică a parametrilor, aceştia se pot conecta pe rînd la blocul 36 (trasator). Pentru a afişa numeric parametrii redaţi, aceştia sînt din nou daşi prin lanţul 1,3... 11, 12, 13, 14, 15.

In realitate, la afişarea semnului (blocul16) nu e numai capsula de afişaj cum rezultă din figură ci o schemă ceva mai complexă, care conţine un bistabil de memorare a semnului parametrului curent şi o logică combinaţională ce asigură selectarea semnului înregistrat sau redat, al parametrului curent

selectat (multiplexat) sau distribuit (demultiplexat) la momentul de timp cînd aceasta este la mijlocul perioadelor de tranziţie. Nu ne propunem să analizăm aici aceste procese.

8.2.3. Principiul de generare al comenzilor. Segmentarea unităţii de comandă - UC

Datorită multitudinii de blocuri funcţionale(36 blocuri), datorită celor două regimuri distincte (înregistrare, redare), precum şi datorită faptului că există mai multe funcţii ce trebuiesc efectuate simultan şi în corelare strînsă una cu alta, UC este deosebit de complexă.

Pentru aceste motive, UC a fost segmentată în 5 părţi: I, H, T, A, R - a căror misiune de ansamblu a fost enunţată în paragraful 8.2.

Aceste 5 părţi sînt de fapt unităţi de comandă, fiecare cu orologiu propriu, avînd doar generatorul de tact unic, pentru a asigura un sincronism perfect între ele. Toate semnalele de dialog cu exteriorul, precum şi semnalele de răspuns interne, de durată sau întîrziere aleatorie, sînt sincronizate cu tactul sistemului, pentru a evita fenomene de hazard dinamic.

La alegerea principiului de funcţionare a unui astfel de orologiu, se putea merge pe un orologiu cu număr fix de impulsuri ce apar succesiv la diferitele ieşiri sau un orologiu la care anumite impulsuri sau grupe de impulsuri se repetă în funcţie de anumite condiţii externe , fig.8.27a, 8.27b. Dar datorită faptului că aceste unităţi de comandă (1,...,H) sînt condiţionate de mai multe semnale de stare interne sau externe, ce pot modifica durata sau tipul ciclului orologiului, s-a ales un astfel de principiu care permite caprice impuls de orologiu să apară în orice moment. De asemenea un impuls de orologiu poate dura nu o perioadă de tact, ci oricîte (similar cu starea TW la microprocesorul INTEL 8080). In plus, ciclul orologiului poate avea durată variabilă în funcţie de combinaţiile semnalelor de condiţionare. In principiu, un astfel de orologiu este prezentat în fig.8.27c.

In continuare, se descriu cele 5 UC, prin

- organigrame
- programe în limbajul hardware AHPL
- tabele cu microoperaţiile (comenzile)

- limbaj obişnuit (descriere prin cuvinte).

8.2.3.1. Unitatea de comandă I, este de fapt un supervizor al proceselor majore care au loc la înregistrare. Organigrama de funcţionare este prezentată în fig.8.28a, şi 8.28b împreună cu blocurile cu borne care sînt în legătură cu această UC. Descrierea AHPL este dată în continuare.

I1.⟶$\overline{(SYN(start)+SYN(porn).\overline{b_1\ porn})}$/(1); rsel← -1.

I2. M_1^*←1; $M_{12,5}$←1; ican = 1.

I3. (SYN($\overline{becan}$)/(3).

I4. pa - rt = SYN (start)

I5.⟶(SYN($M_{12,5}$))/(5).

I6.⟶($\overline{selh}$, selh)/(7,1).

I7. dep - rsel = 1;⟶(2).

unde: - I1...I7 reprezintă impulsurilor de orologiu
- funcţia SYN(X) reprezintă ieşirea unui bistabil de sincronizare de nivel (tip D), la a cărui intrare se aduce funcţia X.
- ⟶ reprezintă faptul că urmează un salt necondiţionat dacă nu există "/".
- (....)/(....) - reprezintă faptul că urmează un salt condiţionat, la care condiţiile de salt sînt în stînga semnului "/", iar destinaţiile sînt în dreapta semnului "/".
- restul notaţiilor reprezintă semnale de comandă ce se generează condiţionat sau nu, la diverse impulsuri de orologiu.

Urmărim fig.8.28 (blocurile cu borne), unde toate dreptunghiurile complete reprezintă blocurile funcţionale ale acestei UC, respectiv orologiul I şi logica sa combinaţională, iar jumătăţile de dreptunghiuri reprezintă blocuri funcţionale comandate de către I.

RSEL - este un registru de selecţie a celor 6 parametri. El poate fi încărcat cu 1000 0000, la comanda rsel←1, conţinutul poate fi deplasat (rotit) la comanda dep - rsel.

Ieşirile sale, nefiind semnificative aici, nu sînt tratate, cu excepţia celei mai semnificative; selh (selecţia para-

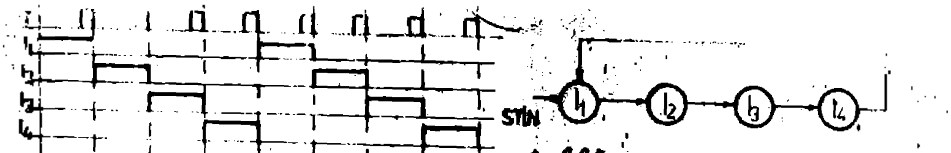


fig 8.27.a

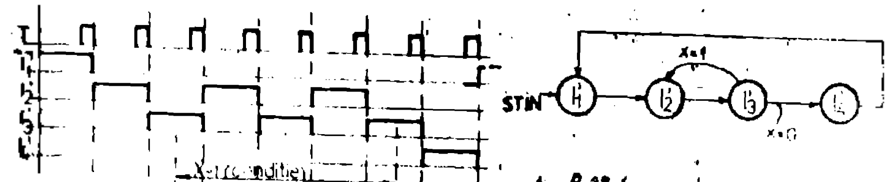


fig 8.27.b

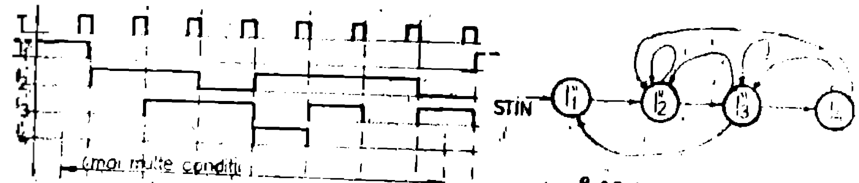
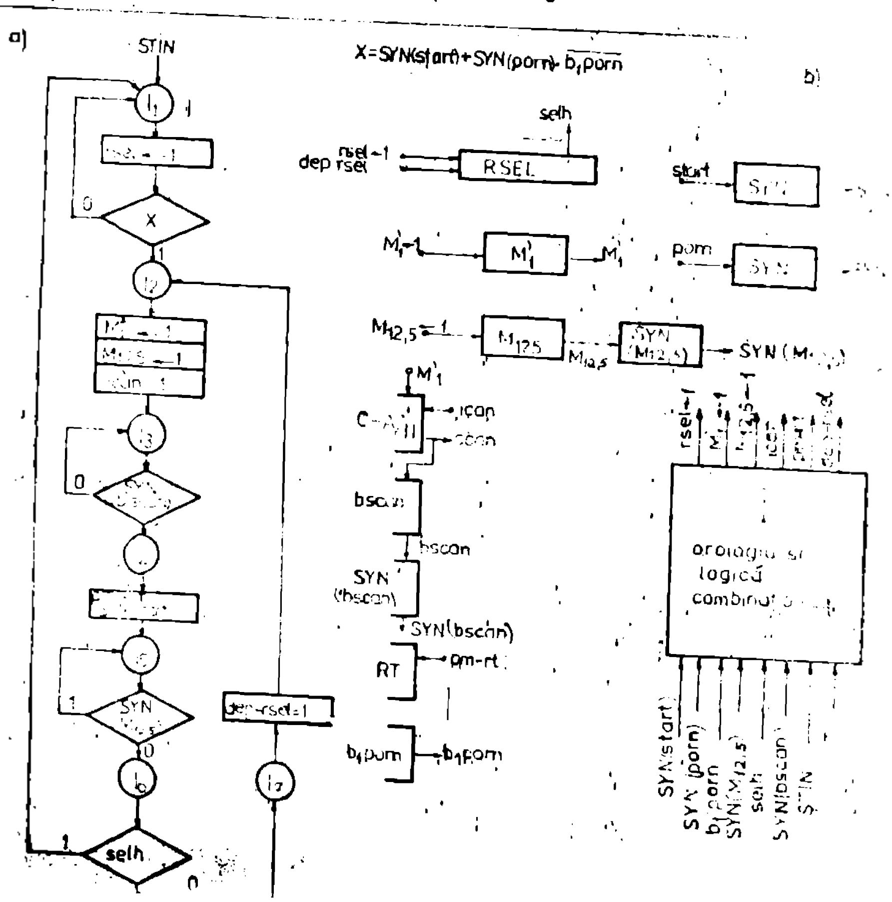


fig. 8.27.c



metrului "timp").

M_1' - este un monostabil cu durata impulsului emis de 1 microsecundă. El este declanşat cu semnalul M_1' 1, şi serveşte convertorului A/N la eşantionare.

$M_{12,5}$ - este un monostabil cu durata impulsului emis de 12,5 ms şi asigură temporizarea la explorarea parametrilor de intrare. Este declanşat cu ajutorul comenzii $M_{12,5}$ 1.

C-A/N - convertor analog-numeric, declanşat cu semnalul incan (şi M_1'); scan este ieşirea sa de stare care anunţă: sfîrşit - conversie - analog - numerică.

bscan - bistabil care memorează semnalul scan

RT - registru tampon (bloc 10, fig.8.26)
încarcă paralel cuvîntul prezentat la intrare, la comanda pm-rt.

$b_1$porn - bistabil care se pune pe "1" la pornirea redării şi stă pe "1" pînă la prima descărcare a RI pe bA_{m_1} (bloc 20, fig.8.26).

start - este semnal de declanşare a înregistrării (de la instalaţia de sudură).

porn - este semnal de declanşare a redării (de la panoul de comandă - PC, fig.8.26).

O descriere tabelară a funcţionării blocului I, este redată în continuare:

O descriere prin cuvinte a proceselor care au loc în unitatea de comandă I, se poate face urmărind organigrama din fig.8.28. sau tabelul 1 (care de fapt reprezintă acelaşi lucru) la fel programul AMPL.

In starea I1 (la impulsul de orologiu I1), se încarcă registrul de selecţie RSEL cu 1000 0000. Dacă X = 0, adică nu a apărut nici semnalul "start", nici semnalul "porn" în condiţia $\overline{b_1 porn}$ = 1, blocul 1 nu este declanşat, adică rămîne în starea de aşteptare I1. Dacă X=1, merge în starea I2 unde se declanşează cele două monostabile amintite şi se comandă începerea unei conversii A/N.

Tranziţiile directe de la un impuls de orologiu la altul, nu sînt specificate explicit.

Starea I3 aşteaptă terminarea conversiei A/N; cînd aceasta s-a produs (bscan=1) se trece mai departe la I4.

In starea I4 se generează comanda pm-rt numai dacă sîntem

UNITATEA DE COMANDA 1

I1	I2	I3	I4	I5	I6	I7
I1 ⟶ : (rsel ⟵ 1) dacă X=1 treci la I2 dacă $\overline{X}$=1 rămîi la I1	I2 ⟶ : (M_1 ⟵ 1) ($M_{12,5}$ ⟵ (ican))	dacă: SYN(bacan)=1 treci la I4 dacă: SYN(bacan)=1 rămîi la I3	I4 ⟶ : (pm-rt)=1 dacă SYN(start)=1	dacă: SYN($M_{12,5}$)=1 rămîi la I5 dacă: SYN($\overline{M_{12,5}}$)=1 treci la I6	dacă $\overline{selh}$=1 treci la I7	I7 ⟶ : (dep-rsel) treci la I2

unde: X = SYN(start) + SYN(porn).$\overline{b_1 porn}$

pe modul de lucru "înregistrare" adică start=1, fiindcă numai atunci RT trebuie încărcat.

In starea I5 se aşteaptă un timp foarte lung (cca 2500 de impulsuri de tact) pînă se termină impulsul $M_{12,5}$, care a fost declanşat cu I2.

In starea I6, dacă selh=1, adică a fost selectat şi ultimul parametru (timpul), se trece la I1 pentru a nu începe o nouă grupă de selecţii de 8 parametrii, dacă "start" şi "porn" între timp nu au devenit ambele "0" -adică s-a încheiat de fapt redarea sau înregistrarea curentă.

Dacă nu s-a selectat ultimul parametru, adică $\overline{selh}$=1, se trece la I7, unde se dă comanda de deplasare a registrului de selecţie (dep-rsel), pentru a se selecta un nou parametru, şi se face salt înapoi la I2 unde se declanşează din nou temporizarea de 12,5 ms şi se porneşte conversia A/N.

Generatorul pentru impulsuri de orologiu I este realizat sub formă de registru cu bistabile D.

Poziţionarea pe "1" a unui bistabil oarecare este definită de ecuaţiile "D" care pot fi scrise după tabelul I, după organigrama din fig.8.28. sau după programul AHPL-I, şi sînt:

$$D_i1 = \overline{I1(SYN(start)+SYN(porn).\overline{b_1porn})} + I6.selh$$

$$D_i2 = I1.(SYN(start)+SYN(porn).\overline{b_1porn})+I7$$

$$D_i3 = I2 + I3(SYN(\overline{bscan}))$$

$$D_i4 = I3(SYN(bscan))$$

$$D_i5 = I4(SYN(M_{12,5}))$$

$$D_i6 = I5(SYN(\overline{M_{12,5}}))$$

$$D_i7 = I6.selh$$

Ecuaţiile funcţiilor sau semnalelor de comandă sînt:

$rsel \leftarrow 1 = I1$

$b_1 \leftarrow 1 = I2$

$M_{12,5} \overline{\leftarrow} 1 = I2$

$icon = I2$

$pm-st = I4.SYN(start)$

$dep-rsel = I7$

Pe baza acestor ecuaţii, sinteza schemei nu prezintă nici o problemă. Ea este dată în fig.8.29.

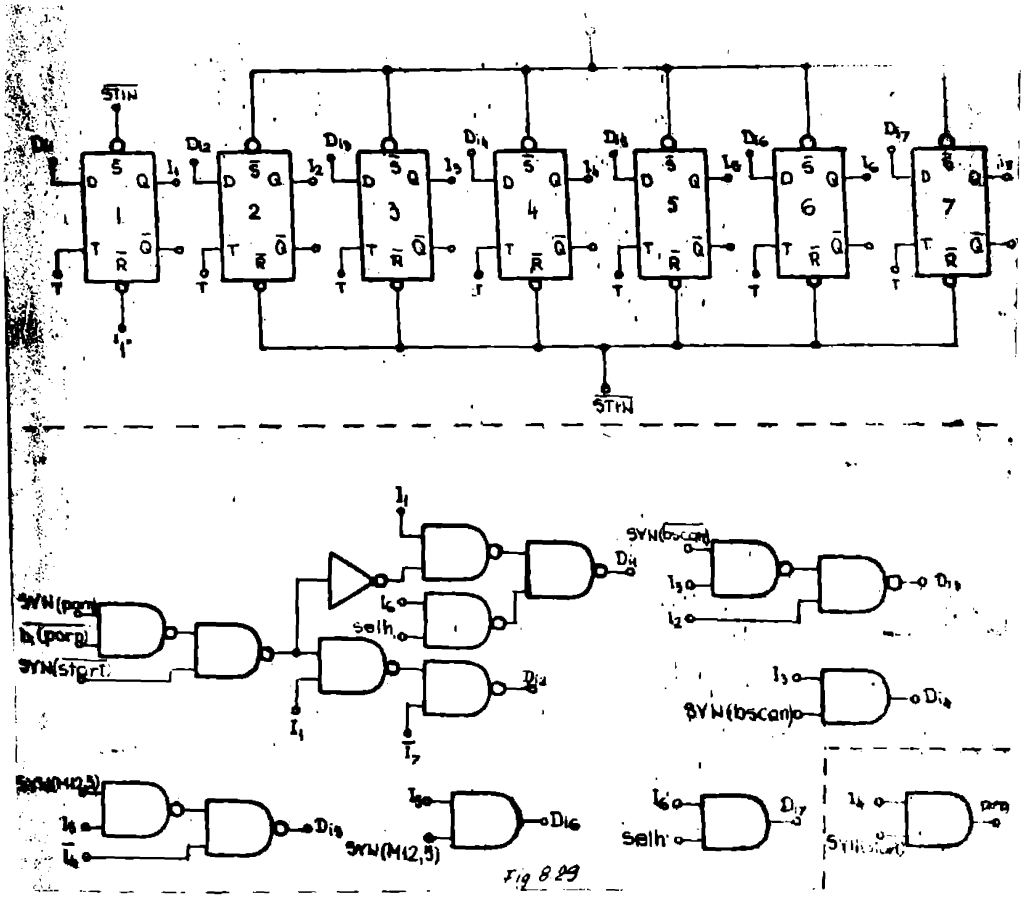


Fig 8.29

8.2.3.2. Unitatea de control 1, este un supervizor al procesoarelor care au loc la redare. Blocurile cu borne care sînt cuplate cu R şi organigrama de funcţionare, sînt prezentate în fig.8.30. Descrierea sa este dată în continuare.

R1. $\longrightarrow$ (SYN($\overline{porn}$))/(1); rdis $\longleftarrow$ 1.

R2. cit = $\overline{b_1 porn}$; cit-cos = $b_1 porn$.

R3. $\longrightarrow$ ($\overline{dval}$)/(3).

R4. $\alpha^e_1 \longleftarrow 1$; $\alpha^e_{12,5} \longleftarrow 1$; $\longrightarrow$ (SYN($\alpha^o_{12,5}$))/(4)

R5. $\longrightarrow$ ($\overline{dish}$, dish)/(6,1)

R6. dep-rdis = 1; $\longrightarrow$ (2).

In fig.8.30 RDIS este un registru de distribuţie a celor

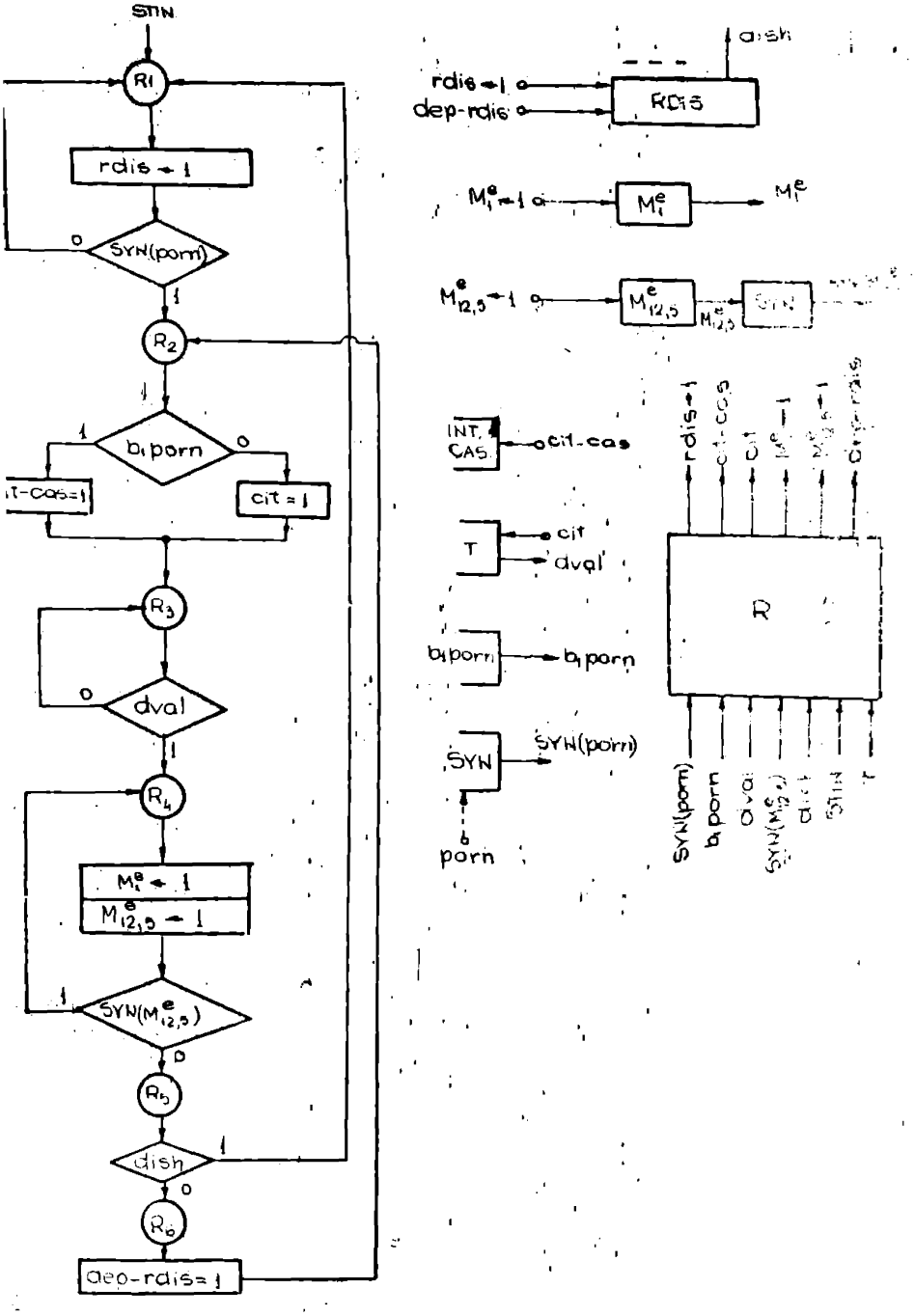

STIN
R1
rdis ← 1
SYN(porti)
R2
bi porn
cit = 1
R3
dval
R4
SYN(M^e_12,5)
R5
dish
R6
dep-rdis = 1
rdis ← 1
dep-rdis
RDIS
INT. CAS
cit-cas
T
cit
dval
bi porn
SYN
SYN(porti)
porn
R

FIG. 8.30.

8 parametri de ieşire. El poate fi încărcat cu 1000 0000 la comanda rdis←1, conţinutul poate fi deplasat (rotit) la comanda dep-rdis. Dintre ieşirile sale, este notată numai cea care ne interesează aici /"dish" (distribuţia ultimului parametru-timpul).

M_1^e - monostabil (1 microsecundă) necesar convertorului N/A, declanşat cu $M_1^e \leftarrow 1$.

$M_{12,5}^e$ - monostabil care asigură temporizarea la distribuirea parametrilor de ieşire, declanşat cu $M_{12,5}^e$ 1.

INT.CAS - interfaţă cu casetă UCM 101-2.

T - unitatea de comandă T (va fi descrisă mai jos)

b_1 porn - a fost descris mai sus

SYN(porn) - a fost descris mai sus

O descriere tabelară a funcţionării blocului R, este dată în continuare:

O descriere prin cuvinte a proceselor care au loc la declanşarea unităţii de comandă t, se dă în continuare.

In starea R1 se încercă registrul de distribuţie RDIS cu 1000 0000, pentru distribuirea primului parametru redat.

Se aşteaptă în starea R1 pînă cînd semnalul "porn" trece pe "1" adică se cere de la panoul de comandă operaţia REDARE.

In starea R2 se dă comanda "cit" - pentru încărcarea tamponului RT (blocul 10 - fig.6.26) cu date din RAM_2 (bloc 21) sau dacă sîntem în primul ciclu de la declanşarea redării, cînd $RAM_{1,2}$ sînt goale se dă comanda "cit-cas" pentru citire de pe casetă şi încărcare $RAM_{1,2}$. Decizia care din aceste două comenzi este dată, e funcţie de $b_1$porn.

In starea R3 se aşteaptă semnalul "dval" - date valide pe 9 biţi paralel în registrul tampon RT, generat de UC-T.

In starea R4 se declanşează două monostabile (cu primul front al lui R4), M_1^e - pentru eşantionare la redare, şi $M_{12,5}^e$ - pentru temporizare la redare.

Tot în această stare se aşteaptă cca 2500 perioade de tact pînă se termină impulsul $M_{12,5}^e$.

In starea R5 se cercetează conţinutul RDIS. Dacă avem "1" la ieşirea dish - distribuirea ultimului parametru - timpul, atunci se trece din nou la R1, pentru a începe un nou set de distribuţii, dacă redarea e activă (porn=1); sau se trece la R6 dacă dish=1, unde se face deplasarea în registrul de distribuţie RDIS, pentru că nu sîntem încă la ultimul parametru. Apoi se trece la R2, unde se va comanda "cit", etc.

UNITATEA DE COMANDA R

R1	R2	R3	R4	R5	R6
R1 $\longrightarrow$: (rdis $\longleftarrow$ 1)	R2 $\longrightarrow$: cit=1 dacă	dacă:dval=1 treci la R4	R4 $\longrightarrow$: ($M_1^e \longleftarrow 1$) $M_{12,5}^e \longleftarrow 1$)	dacă $\overline{dish}$=1 treci la R6	R6 $\longrightarrow$: (dep-rdis)
dacă: SYN($\overline{porn}$)=1 rămîi la R1	$\overline{b_1 porn}$=1 cit-cqs=1 dacă $b_1$porn=1	dacă:$\overline{dval}$=1 rămîi la R3	dacă: SYN($M_{12,5}^e$)=1 rămîi la R4	dacă dish=1 treci la R1	treci la R2
dacă: SYN(porn)=1 treci la R2			dacă: SYN($\overline{M_{12,5}^e}$)=1 treci la R5		

Ecuațiile D sînt următoarele:

$D_{r1} = R_1.SYN(\overline{porn}) + R_5.dish$

$D_{r2} = R_1.SYN(porn) + R_6$

$D_{r3} = R_2 + R_3.dval$

$D_{r4} = R_3.dval + R_4.SYN(M^e_{12,5})$

$D_{r5} = R_4.SYN(\overline{M^e_{12,5}})$

$D_{r6} = R_5.\overline{cish}$

Ecuațiile semnalelor de comandă sînt:

$rdis \leftarrow 1 = R1$

$cit = R2.\overline{b_1 porn}$

$cit\text{-}cas = R2.b_1 porn$

$M^e_1 \leftarrow 1 = R4$

$M^e_{12,5} \leftarrow 1 = R4$

$dep\text{-}rdis = R6.$

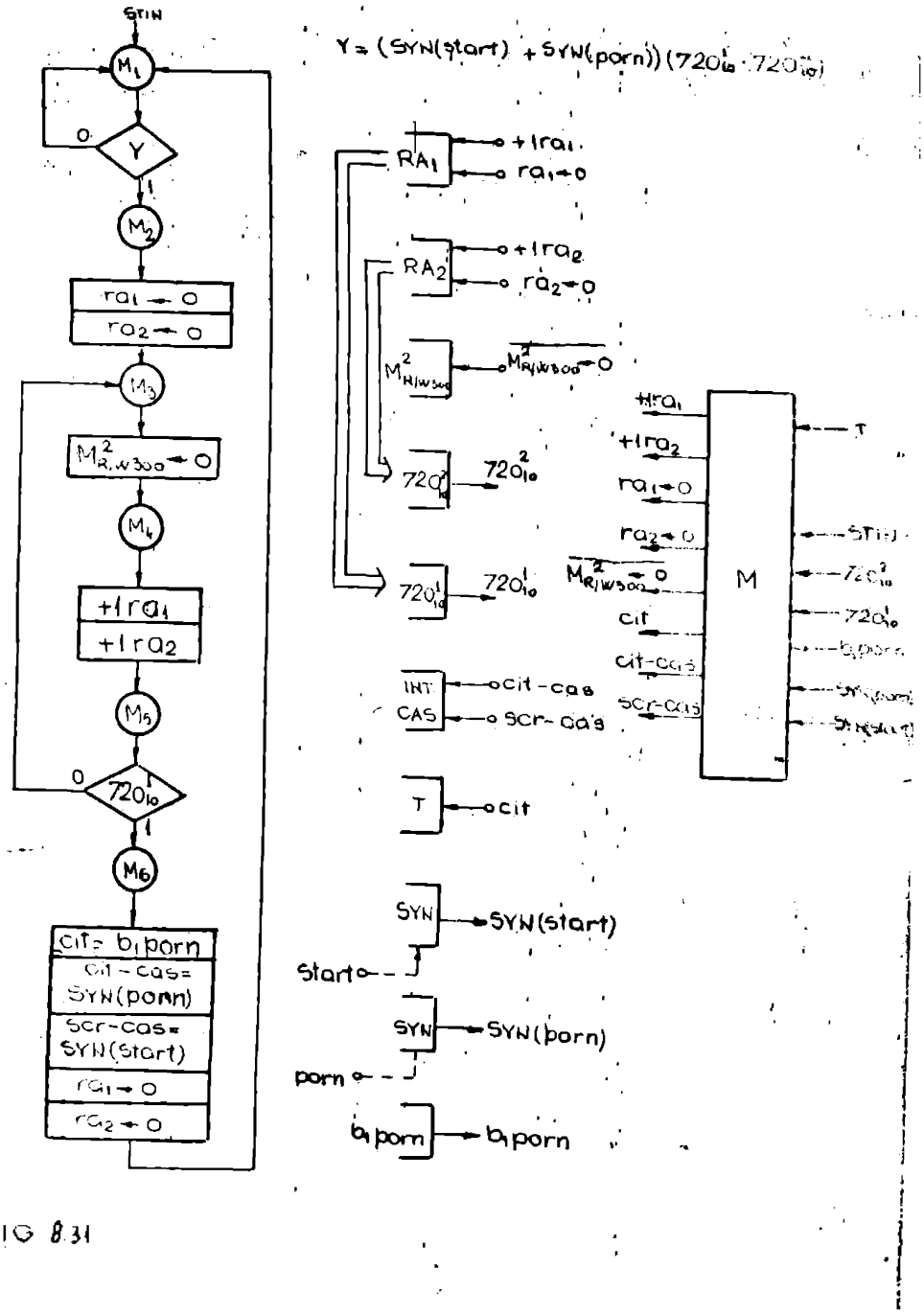
Nu considerăm necesar a fi prezentată schema rezultată din aceste ecuații și nici la celelalte UC - care urmează.

5.2.3.3. Unitatea de comandă M_3 cercetează atît la înregistrare cît și la redare faptul dacă RAM_1 este plin de date (720^1_{10}= =1) și RAM_2 gol (720^2_{10}= 1) aduce la "0" registrele de adrese ale celor două RAM-uri declanșează ciclu de citire la RAM_1 și de scriere la RAM_2 copiază un bit din RAM_1 în RAM_2, avansează cu o unitate adresele, iar dacă nu s-a ajuns la adresa finală în RAM_1(720^1_{10} = 1), revine la copiere bit-cu-bit.

Dacă s-a ajuns la adresa finală (720^1_{10}=1), adică s-a încheiat copierea conținutului lui RAM_1 în RAM_2, atunci trece din nou în starea de așteptare, dă comanda "cit" dacă $b_1$porn=1, și în plus la înregistrare se dă comanda "scr-cas", iar la redare- "cit-cas", spre interfața casetei pentru a scrie un bloc de date din RAM_2 pe casetă, sau de a citi de pe casetă un bloc de date în RAM_1.

In fig.5.31 sînt prezentate: blocurile cu borne, și organigrama de funcționare.

In continuare sînt date: descrierea prin tabel, descrierea prin limbaj AHPL și ecuațiile "D" și ale semnalelor de comandă.


STIN
M1
Y
M2
ra1 ← 0
ra2 ← 0
M3
M2R,W300 ← 0
M4
+1ra1
+1ra2
M5
720¹10
M6
cit ← b1porn
cit-cas ← SYN(porn)
scr-cas ← SYN(start)
rG1 ← 0
rG2 ← 0
Y = (SYN(start) + SYN(porn))(720¹10 · 720²10)
RA1
+1ra1
ra1 ← 0
RA2
+1ra2
ra2 ← 0
M2R,W300
M2R,W300 ← 0
720²10
720¹10
INT
CAS
cit-cas
scr-cas
T
cit
SYN
SYN(start)
start
SYN
SYN(porn)
porn
b1porn
b1porn
M
T
STIN
720²10
720¹10
b1porn
cit

IG 8.31

M1. $\longrightarrow((SYN(start)+SYN(porn))(720^1_{10}.720^2_{10}))/(1)$.

M2. $re_1, re_2 \leftarrow 0$.

M3. $\overline{MR/W}^2\overline{300} \leftarrow 0 = 1$.

M4. $+1re_1$, $+1re_2 = 1$.

M5. $\longrightarrow(720^1_{10}, \overline{720^1_{10}})/(6,3)$.

M6. $\longrightarrow(1)$; $re_1, re_2 \leftarrow 0$; $cit=b_1porn$;
cit-cas = SYN(porn); scr-cas = SYN(start)

$D_{m1} = M1.\overline{Y} + M6$

$D_{m2} = M1.Y$

$D_{m3} = M2 + M5\ (720^1_{10})$

$D_{m4} = M3$

$D_{m5} = M4$

$D_{m6} = M5.720^1_{10}$

$re_1 \leftarrow 0 = M2 + M6$

$re_2 \leftarrow 0 = M2 + M6$

$M^2_{R/W\ 300} \leftarrow 0 = M3$

$+ 1\ re_1 = M4$

$+ 1\ re_2 = M4$

$cit = M6.b_1porn$

cit-cas = M6.SYN(porn)

scr-cas = M6.SYN(start)

Considerăm suficient cît s-a prezentat pînă acum despre această unitate de comandă, întrucît toate semnalele şi blocurile în cauză fie au mai fost prezentate, fie semnificaţiile lor sînt clare (ex: $+ 1re_1$, cit, etc.).

UNITATEA DE COMANDA M

M1	M2	M3	M4	M5	M6
dacă: Y=1, treci la M2	M2 $\longrightarrow$: ($ra_1 \leftarrow 0$) ($ra_2 \leftarrow 0$)	M3 $\longrightarrow$: ($\overline{M^2_{R/W\ 300}} \leftarrow 0$)	M4 $\longrightarrow$: ($+1ra_1$) ($+1ra_2$)	dacă: $720^1_{10}=1$, treci la M6	M6 $\longrightarrow$: (cit=1, dacă: $b_1$porn=1) (cit-cas=1, dacă SYN(porn)=1 (scr-cas=1, dacă SYN(start)=1)
dacă: Y=0, treci la M1 (rămîi)				dacă: $720^1_{10}=0$, treci la M3.	treci la M1

8.2.3.4. Unitatea de comandă T, dirijează încărcarea şi descărcarea serie sau paralel a registrului tampon RT - blocul 10 fig.8.26. Această "UC", funcţionează şi la înregistrare (SEM(start)=1) şi la redare (SEM(porn)=1).

Într-unul din aceste două moduri de lucru, T aşteaptă unul din semnalele rm-rt sau cit, amintite mai sus. Întrucît RT este pe 9 biţi, s-a prevăzut un numărător N9 pentru a număra deplasările în RI şi un bistabil b_n9 pentru a autoriza numărarea în N9. RT generează cifre binare serie spre RAM_1, pe calea 10,19,20 în fig.8.26. Deci la RAM_1 trebuie declanşat ciclu de scriere cu monostabilul al 1/#500 (la înregistrare).

La redare, RT se încarcă serie cu date de pe casetă pe calea: 29,28,19,20,21,10 - fig.8.26.

Tot la redare, un cuvînt de 9 biţi asamblat în RT este semnalizat cu semnalul dval - menţionat mai sus.

Bistabilul "$b_1$porn" despre care s-a mai discutat este anulat tot în UC-1.

Semnalele noi care apar aici sînt:

depl-rt; deplasare a conţinutului RT

-1N9; scădere conţinut N9

$b_n9 \leftarrow 1$; punere pe "1" a bistabilului "N9".

N9 $\leftarrow$ "0"; încărcare N9 cu 1000.

Întocmai ca la celelalte unităţi de comandă, se dă şi aici organigrama de funcţionare şi blocurile cu borne - care sînt comandate de UC-1, în fig.8.32.

În continuare sînt date: tabelul de descriere a funcţionării UC-1, ecuaţiile D şi ecuaţiile comenzilor.

t1. $\rightarrow (\overline{s})/(1)$; N9 $\leftarrow$ "0"; $b_n9 \leftarrow 1$.

t2. [illegible] $\leftarrow$ 0 SEM(start); -1N9 = 1.

t3. depl-rt=1; $+1rm_1$=SEM(start); $+1rm_2$=SEM(porn)

t4. $\rightarrow (b_n9, \overline{b_r9})/(2,5)$

t5. dval = SEM(porn); $\rightarrow (1)$; b_1 porn $\leftarrow$ 0.

D_{b_1} = 2.t1 + t5

D_{b_n} = 2.t1 + t4.b_n9

D_{t_1} = t2

Dt_4 = T3

Dt_5 = T4.$\overline{b_n 9}$

$+lrs_1$ = T3.SYN(start)

$+lre_2$ = T3.SYN(porn)

N9 ← "b" = T1

$b_n 9 \leftarrow 1$ = T1

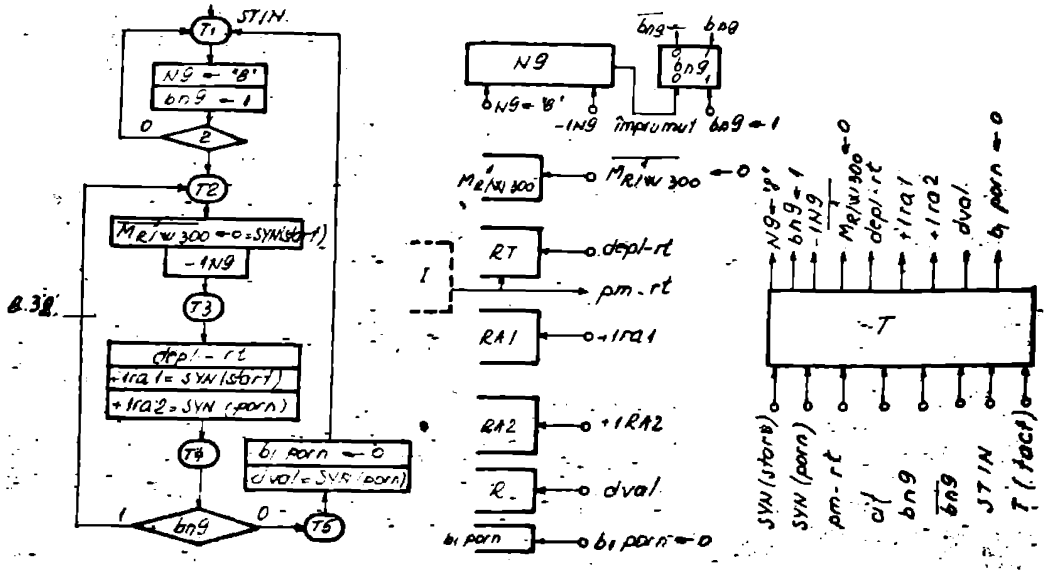
$\overline{M^1/A}$ 300 ← 0 = T2.SYN(start)

-1N9 = T2

depl-rt = T3

dval = T5.SYN(porn)

$b_1$porn ← 0 = T5



În continuare, va fi prezentată o unitate de comandă mai complexă, cea care la sfîrșitul conversiei A/N, preia cuvîntul binar, îi efectuează corecția numerică de scară, îl convertește din binar în BCD, după o metodă "serie" și îl codifică pentru afișajul pe 3 cifre zecimale realizate din 7 segmente.

UNITATEA DE COMANDA T

T1	T2	T3	T4	T5
T1 ⟶ : (N9 ← "8") (bn9 ← 1) --- dacă Z=0: rămîi la T1 --- dacă Z=1: mergi la T2	T2 ⟶ : (- 1N9) --- ($M^{1}_{R/W}$ 300 ← 0), dacă SYN(start)=1	T3 ⟶ : (depl-rt) (+1ra_1,dacă SYN(start)=1) (+1ra_2, dacă SYN(porn) =1)	dacă: bn9=1,treci la T2 --- dacă: bn9 = 1, treci la T5	T5 ⟶ : ($b_1$porn ← 0) (dval, dacă SYN(porn) = 1) --- treci la T1

Z = SYN(start).pm-rt+SYN(porn).cit

8.2.3.5. Unitatea de comandă M, efectuează operațiile amintite mai sus, este suficient de complexă și pentru acest motiv, vor fi prezentate mai multe considerații teoretice, după cum urmează.

Mărimile fizice care se afișează sînt: 3 curenți, două tensiuni, două viteze și timpul. Toți acești parametri, care apar la intrare, sînt reprezentanți prin tensiuni analogice unificate cuprinse între 0 - 10 V.

Curenții sînt cuprinși între 0 - 1000 A.

Tensiunile sînt cuprinse între 0 - 100 V.

Vitezele sînt cuprinse între 0 - 2 m/min (200 cm/min).

Timpul, în forma binară, este cuprins între 0 - 250 și trebuie afișat nemodificat.

Fiecărui parametru îi este atașat un semn, inclusiv timpului, pentru a marca grupele de impulsuri de timp, adică (0 + - 250) (0 + + 250) (0 + -250), etc.

Pentru afișarea tensiunii unificate (0 + + 10 V), pe 3 ranguri zecimale, reprezentînd mărimi fizice diferite, la care valoarea maximă este 1000 sau 100 sau 200, trebuie făcută o modificare de scară pentru fiecare parametru sau mărime fizică.

De exemplu: tensiunea arcului electric variază în domeniul 0 - 100 V; pe cele 8 ranguri binare ale C-A/N se pot obține valori numerice cuprinse între 0 + 255 ; cu alte cuvinte, la 100 V ar corespunde cifra 255 la ieșirea C-A/N. Dar trebuie afișat de fapt 100.

Deci:

X	X	Z
255	100	+10
0	0	0

unde:

X = cifra obținută la ieșirea C-A/N

Y = mărime fizică de afișat

Z = tensiune unificată de la traductorul înălțimii de sudare.

A se efectua corecția sau modificarea de scară în acest caz, numai pe cale numerică, este foarte dificil, și se procedează astfel:

- se efectuează o corecție parțială pe cale analogică,

prin divizarea potențiometrică.

- se efectuează o corecție pe cale numerică, prin înmulțiri sau împărțiri succesive la 2.

Mai întîi se divide mărimea Z, cu un astfel de raport încît mărimea X să fie un multiplu sau submultiplu de 2 al mărimii Y.

Astfel dacă considerăm:

X_1	Y_1	Z_1
200	100	$\frac{10}{255}\cdot 200$

deci am avut: 255 ------- 10

și avem: 200 ------- $\frac{10}{255}\cdot 200$

iar $\frac{Z_1}{Z} = \frac{\frac{10}{255}\cdot 200}{10} = \frac{200}{255} = 0{,}787$ - raportul de divizare

Acum: $Y_1 = X_{1/2}$, care reprezintă tocmai relația corecției numerice de scară; adică pentru obținerea cifrelor care trebuiesc afișate, trebuie făcută o împărțire la 2 a cuvîntului binar de la ieșirea C-A/N.

In continuare, pentru curentul de sudare, avem:

X'	Y'	Z'
255	1000	+10
0	0	0

notațiile fiind similare ca mai sus.
Și în continuare:

X_1'	Y_1'	Z_1'
250	1000	$\frac{10}{255}\cdot 250$

de aici: $Z_1' = \frac{\frac{10}{255}\cdot 250}{} = \frac{250}{255} = 0{,}982$ - raportul de divizare

Acum: $Y_1' = X_1'.4$, care reprezintă relația corecției numerice de scară.

Pentru viteza de avans a tractorului de sudare avem:

X"	Y"	Z"
255	200	+10
0	0	0

şi:

X_i''	Y_i''	Z_i''
200	200	$\frac{10}{255}\cdot 200$

de aici:

$$\frac{Z_i''}{Z''} = \frac{\frac{10}{255}\cdot 200}{10} = \frac{200}{255} = 0{,}787$$ - raportul de divizare.

se observă că: $Y_i'' = X_i''$, deci nu mai trebuie efectuată nici un fel de corecţie numerică de scară.

La parametrul timp, nu mai trebuie efectuată nici măcar corecţia analogică de scară.

Pentru realizarea acestor corecţii se utilizează un registru de deplasare RD, care este încărcat paralel cu cuvîntul de la ieşirea C-A/N, iar apoi în acest registru se vor face: o deplasare la dreapta, pentru împărţire cu 2, la tensiuni, două deplasări la stînga pentru înmulţire cu 4, la curenţi, şi nici o operaţie la viteze sau la parametrul timp.

Pentru a converti acest cuvînt corectat din binar în BCD (zecimal-codificat-binar) conţinutul RD-după corecţie, se încarcă paralel într-un numărător binar-invers (NBI).

Un alt numărător, zecimal direct (NZ) este adus la zero, şi un bistabil de autorizare a numărării (bn) este adus la "1".

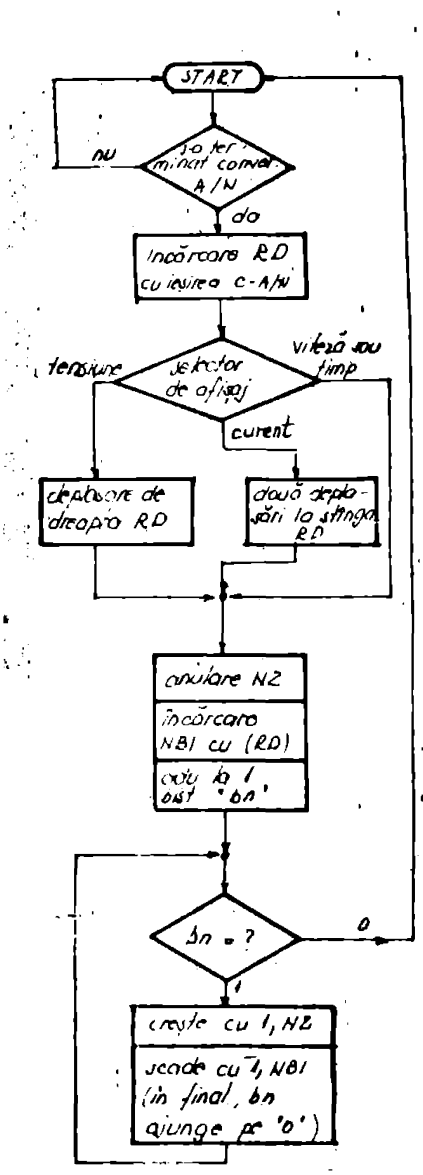
Se numără simultan în ambele numărătoare, pînă cînd NBI are conţinut 0 ... 0 şi la următorul impuls de numărare apare împrumutul de la acest numarător, care aduce la zero "bn". În acest moment conversia este efectuată, şi NZ conţine pe 3.4=12 biţi cele 3 cifre zecimale care trebuiesc afişate. Cu cîte un decodificator BCD-7 segmente, pe fiecare rang zecimal, s-a terminat în sfîrşit problema afişajului.

In fig.6.35 este prezentată organigrama de principiu a acestor fenomene, organigrama de detaliu şi blocurile cu borne, efectate de aceste operaţii.

Elementele nediscutate pînă acum sînt:

- MUX-RD - multiplexor al RD care permite încărcarea RD cu ieşirea C-A/N sau cu propria sa ieşire, conectată deplasat la stînga, pe care nu le poate efectua RD, în funcţie de valoarea semnalului "conex".

- conex - bistabil de comandă MUX-RD.



Organigrama de principiu

Fig. 8.33.a

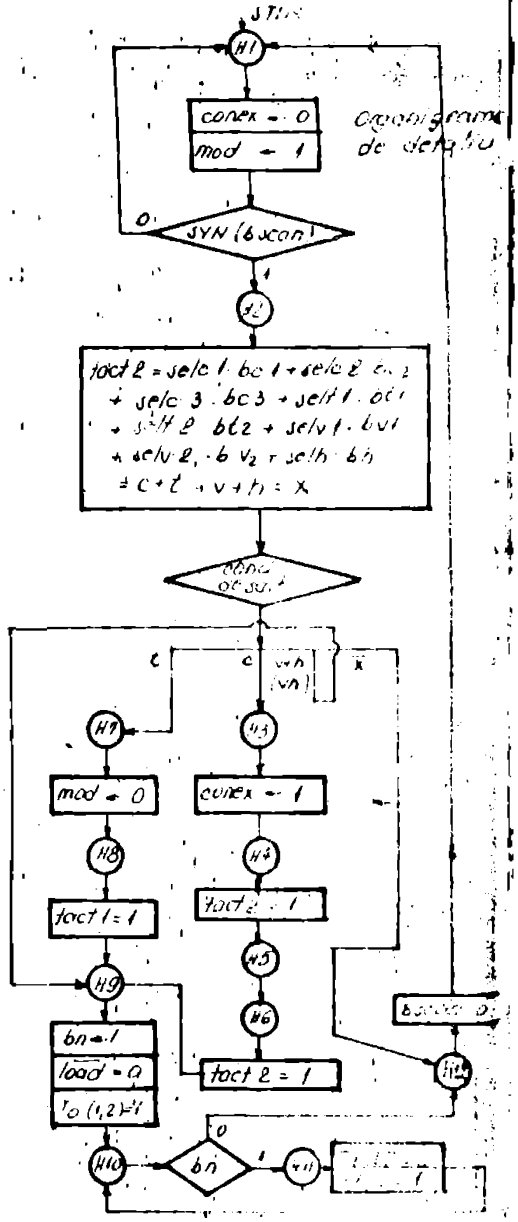


Fig. 8.33.b

- mod - bistabil care stabileşte modul de lucru al lui RD (mod=1; încărcare paralel şi deplasări la stînga; mod=0; deplasări la dreapta).

- RSEL - registru de selecţie a parametrilor (selc1=selecţie curent$_1$,...,selh=selecţie timp).

- selectorul de afişaj - comutator depe PC (fig.6.26 care selectează parametrul vizualizat (bc1 - vizualizat curent$_1$,... bh - vizualizat timpul).

Semnalele de mai jos servesc la:

- tact 1 - deplasare dreapta RD
- tact 2 - încărcare paralel şi deplasare stînga RD
- load - încărcare paralel NBI
- ro(1,2) - anulare NZ.

Funcţionarea UC-H după organigrama din fig.6.33b, este următoarea:

In starea H1, se aduce la "0" bistabilul "conex" pentru a conecta la intrarea RD ieşirea C-A/N prin MUX-RD, se aduce la "1" bistabilul "mod" pentru a pune RD în regim de încărcare paralel. Se aşteaptă în H1 pînă cînd se termină conversia A/N, adică SYN(bscan)=1.

In starea H2 se dă "tact 2" de încărcare paralel a RD numai dacă coincid o pereche de semnale selc$_1$,...etc., cu bc$_1$,...etc., adică RD se va încărca cu ieşirea C-A/N numai dacă de exemplu se cere afişarea unui parametru oarecare, şi tot acel parametru este cel selectat momentan de RSEL.

De exemplu: tact$_2$ = 1 dacă selv$_1$=1 (este selectată momentan viteza 1) şi bv$_1$=1 (este cerută în afişare viteza 1).

Mai departe se poate merge pe patru ramuri:

-t : se va încărca în RD şi în final se va vizualiza o tensiune.

- c : se va încărca şi vizualiza un curent.

- v+h (vh) : se va încărca şi visualiza o viteză sau timpul (la care nu se fac corecţii de scară).

- X : nu se va vizualiza nimic.

Pe ramura t se trece în starea H7 unde se poziţionează bistabilul "mod" pe 0 pentru a permite deplasări la dreapta. In starea H8 se dă comanda "tact$_1$" spre RD pentru a efectua deplasarea la dreapta. In starea H9 se aduce la "1" bistabilul "bn" pentru a autoriza în continuare numărarea în NBI şi NZ.

Cu comanda "load" = 0, se încarcă NBI cu (RD). Cu comanda ro(1,2) = 1, se anulează (NZ).

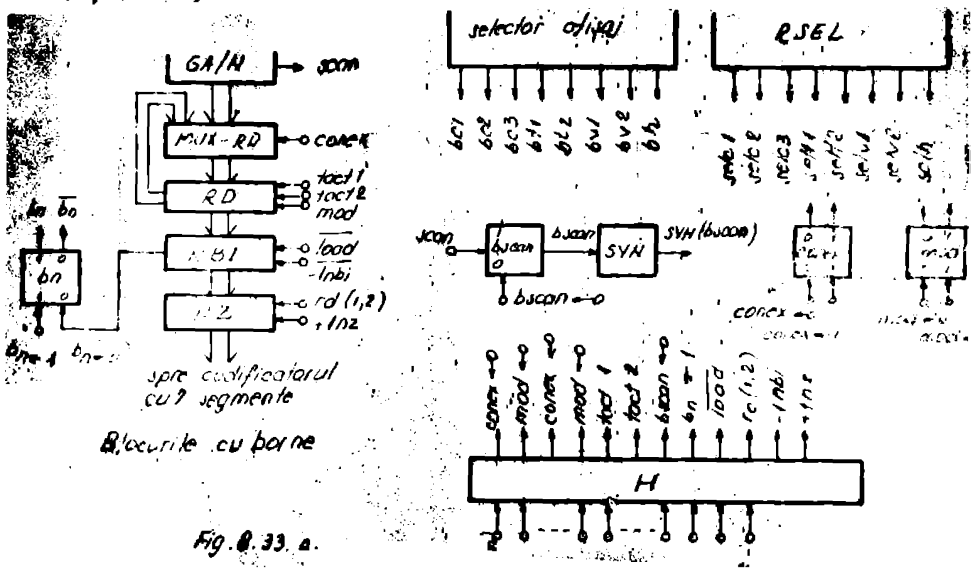


Fig. 8.33. a.

În stările H10 și H11 se face numărarea în NBI și NZ până când bn=0, datorită împrumutului care apare în final la NBI, după care se trece în starea H12. Aici se aduce la "0" bscan și se trece în regim de așteptare în H1. Pe ramura 0, "conex" se poziționează pe "1" pentru a realiza conexiunea ieșirilor RD cu intrările sale, deplasat cu un rang spre stânga prin MUX-RD. (În starea H3).

În starea H4 apare primul tact de deplasare la stânga, "tact$_2$", în H5 se așteaptă terminarea proceselor tranzitorii în urma deplasării, iar în H6 se dă al doilea tact de deplasare la stânga. Se trece la H9.

Deci pe ramura t s-a făcut o împărțire la 2, pe ramura c, o înmulțire cu 4, pe ramura vh nu sînt necesare corecții și se trece direct la starea H9 unde începe conversia binar-BCD, iar pe calea X nefiind necesară nici corecție nici conversie, se trece prin H12 la H1 în regim de așteptare.

În continuare se dau descrierile acestui dispozitiv de comandă, prin limbaj AHPL, prin tabele, și după acestea ecuațiile D și ale semnalelor de comandă.

H1. $\longrightarrow$(SYN($\overline{\text{bscan}}$)/(1);
conex$\longleftarrow$0; mod$\longleftarrow$1.

H2. $tact_2$=$selc_1.bc_1$ +
+$selc_2.bc_2$ +
+$selc_3.bc_3$ +
+$selt_1.bt_1$ +
+$selt_2.bt_2$ +
+ $selv_1.bv_1$+
+$selv_2.bv_2$ +
+selh.bh*c+t+v+h;
$\longrightarrow$(t,c,$\overline{v+h}$,$\overline{\text{vh}}$,$\overline{x}$)/(7,3,9,12)

H3. conex$\longleftarrow$0

H4. $tact_8$ = 1

H5. nul.

H6. $tact_2$ = 1;$\longrightarrow$(9)

H7. mod$\longleftarrow$0.

H8. $tact_1$ = 1

H9. bn$\longleftarrow$1; $\overline{\text{load}}$ = 0; ro(1,2) = 1.

H10.$\longrightarrow$($\overline{\text{bn}}$)/(12).

H11.$\longrightarrow$(10); - $\overline{\text{inbi}}$ = 0; +1.z = 1.

H12. b scan$\longleftarrow$0;$\longrightarrow$(1).

ecuatiile D:

D_{h1} = H1.SYN($\overline{\text{bscan}}$) + H12

D_{h2} = H1.SYN(bscan)

D_{h3} = H2.c

D_{h4} = H3

D_{h5} = H4

D_{h6} = H5

D_{h7} = H2.t

D_{h8} = H7

D_{h9} = H8+H6

D_{h10}=H9+H11

D_{h11}=H10.bn

D_{h12}=H2.x+H10.bn

UNITATEA DE COMANDĂ H

H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12
H1 ⟶: (conex ⟵ 0) (mod ⟵ 1)	H2 ⟶: ($tact_2$=1 dacă: X=1)	H3 ⟶: (conex ⟵ 0)	H4 ⟶: ($tact_2$)	-	H6 ⟶: ($tact_2$)	H7 ⟶: (mod ⟵ 0)	H8 ⟶: ($tact_1$)	H9 ⟶: (bn ⟵ 1) (ro(1,2))	dacă: bn=0 treci la H12	H11 ⟶: (+1nx)	H12 ⟶: (bscan ⟵ 0)
dacă: SYN(bscan)= =1 treci la H2	dacă t=1 treci la H7				treci la H9			H9 ⟶: (load)	dacă: bn=1 treci la H11	H11 ⟶: (-1nbi)	treci la H11
dacă: SYN($\overline{bscan}$)= =0 rămîi la H1	dacă c=1 treci la H3									treci la H10	
	dacă vh=1 treci la H9										
	dacă X=1 treci la H12										

$selc_1 \cdot bc_1 + selc_2 \cdot bc_2 + selc_3 \cdot bc_3 + selt_1 \cdot bt_1 + selt_2 \cdot bt_2 + selv_1 bv_1 + selv_2 \cdot bv_2 + selh \cdot bh = t+c+v+h = t+c+vh = X$

c t v h

ecuaţiile semnalelor de comandă:

conex ⟵ 0 = M1
mod ⟵ 1 = M1
$tact_2$ = M2.x
conex ⟵ 1 = M5
mod ⟵ 0 = M7
$tact_1$ = M8
b_n ⟵ 1 = M9
$\overline{load}$ = $\overline{M9}$
ro(1,2) = M9
- lnbi = $\overline{M11}$
+ lnz = M11
bscam ⟵ 0 = M12

Orientativ, menţionăm că această UC este realizată cu cca 20 de circuite integrate, de provenienţă indigenă.

3.2.4. Unificarea ecuaţiilor semnalelor de comandă

Parcurgînd listele de ecuaţii ale semnalelor de comandă generate cu ajutorul celor 5 orologii (unităţi de comandă), se observă că există semnale de comandă cu acelaşi nume în mai multe UC.

Ecuaţia finală a unui astfel de semnal se obţine efectuînd suma logică a termenilor ce corespund tuturor apariţiilor semnalului respectiv în listele de ecuaţii.

Termenii care provin din interfaţa casetei magnetice vor purta sufixul-cas, şi vor apare în mai multe ecuaţii, dar nu vom analiza modul de apariţie a lor, pentru că nu ne-am propus să analizăm aici, interfaţa casetei.

Se dau mai jos, ecuaţiile unificate ale semnalelor de comandă ce au mai mult de o apariţie.

cit = T2.$\overline{b_1 porn}$ + M6.$b_1$porn
cit-cas=R2.$b_1$porn+M6.SYN(porn)
rs_1 ⟵ 0 = M2+M6+S1LH
rs_2 ⟵ 0 = M2+M6
$+lrs_1$ = M4+T3.SYN(start)+($+lrs_1$-cas)
$+lrs_2$ = M4+T3.SYN(porn)+($+lrs_2$-cas)
$\overline{M_{R/W\ 300}^{\ 1}}$ ⟵ 0=T2.SYN(start)+($\overline{M_{R/W\ 300}^{\ 1}}$ ⟵ 0 - cas)

De menţionat că semnalul de comandă cit-cas nu face parte

din convenţia /"sufixul-case" şi de fapt reprezintă comanda de citire dată casetei magnetice.

In ecuaţia: $ra_1 \leftarrow 0$, apare termenul STIN pentru că memoria MAA_1 trebuie să pleca de la adresa "0".

Memoria RAM_2 trebuie să pleca de la adresa 720 pentru a semnaliza că este "descărcată", deci va mai apare ecuaţia:

$ra_2 \leftarrow 720 = STIN.$

Prea puţine registre, numărătoare sau bistabile trebuie să pleca de la o anumită valoare (0 sau 1) de aceea semnalul STIN apare atît de rar.

8.2.5. Dialogul dintre interfaţa casetei magnetice şi restul instalaţiei şi caseta propriu-zisă

La înregistrare, datele parcurg următorul traseu:

... C-A/N ⟶ RT ⟶ RAM1 ⟶ RAM2 ⟶ INT.CAS ⟶ CAS

iar la redare, traseul:

CAS ⟶ INT.CAS ⟶ RAM1 ⟶ RAM2 ⟶ RT...

unde: CAS - reprezintă unitatea de casetă magnetică iar restul elementelor au fost deja descrise.

La înregistrare, deci caseta este pornită, se descarcă un bloc de date din RAM_2 pe casetă, caseta este oprită, şi fenomenul se repetă, la fiecare apariţie a comenzii "scr-cas".

La redare, caseta este pornită, se citeşte de pe casetă un bloc de date, care umple MAA_1 pînă la adresa 720, caseta este oprită, şi fenomenul se repetă la fiecare apariţie a comenzii "cit-cas".

Deci caseta funcţionează în regimul: start-stop şi lucrează cu blocuri mici de date : 720 biţi.

Pentru a ilustra modul cum interfaţa casetei asigură aceste operaţii se dau organigramele de principiu din fig.8.34.

8.2.6. Intretesarea (suprapunerea) în timp a funcţionării celor 5 unităţi de comandă şi a casetei cu interfaţa

Datorită faptului că în SIR-HM se petrec mai multe grupe de fenomene simultane, ar fi util de arătat cum se suprapun acestea în timp şi cum se corelează între ele. Acest lucru este ilustrat în fig.8.35.

Analizăm acum fenomenele de la înregistrare, ilustrate în

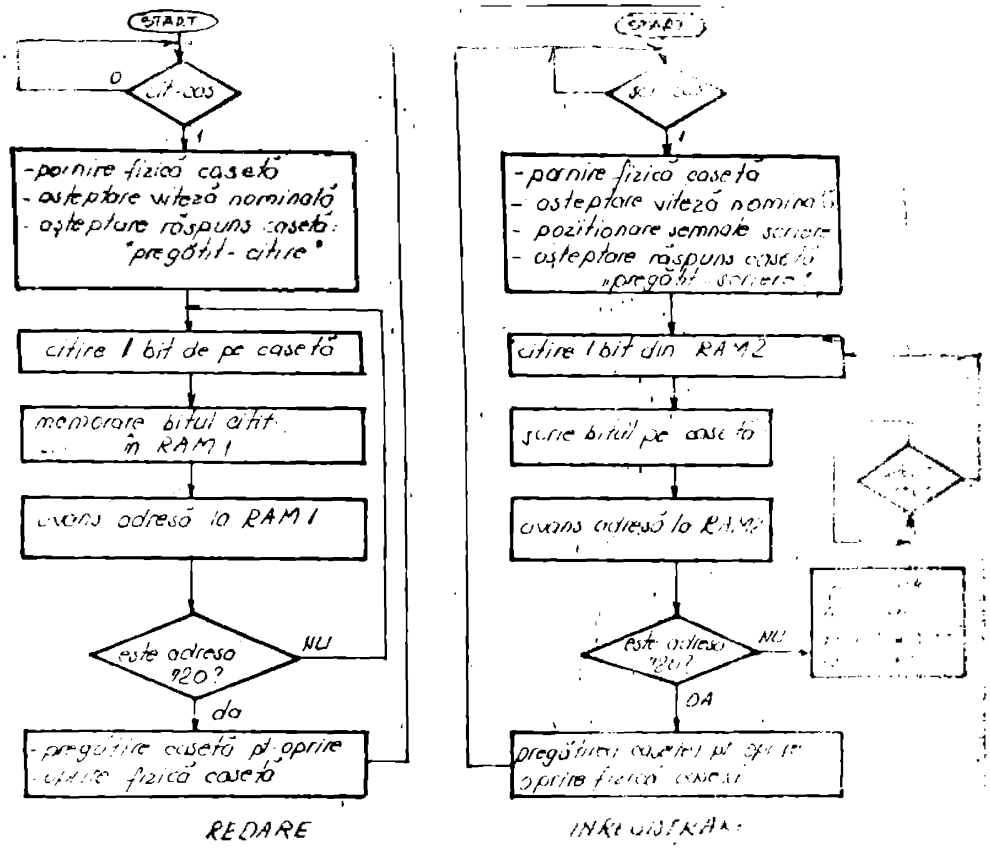
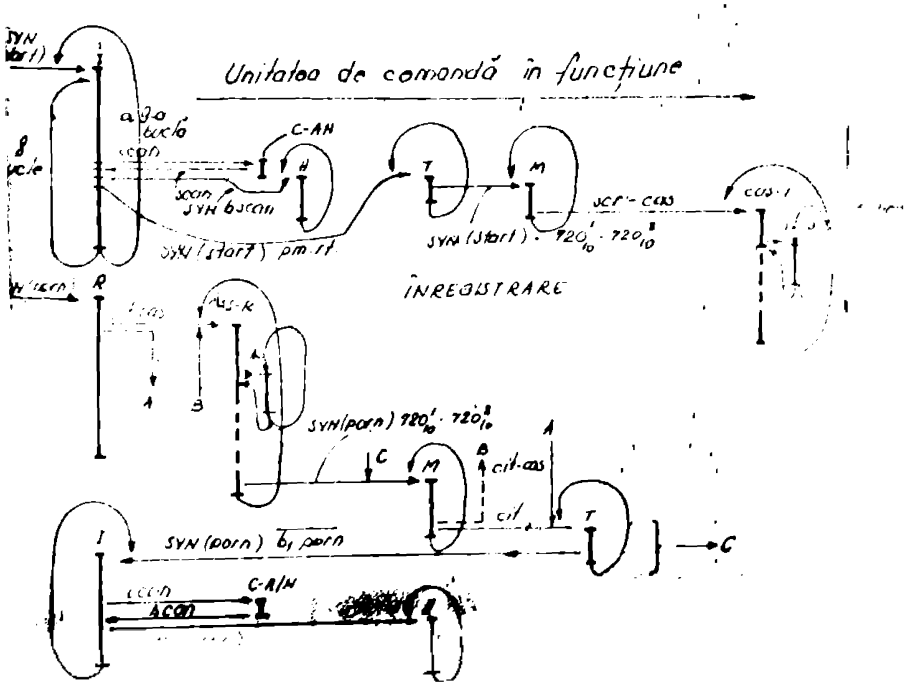

START
cit-cas
-pornire fizică casetă
- așteptare viteză nominală
- așteptare răspuns casetă:
"pregătit - citire"
citire 1 bit de pe casetă
memorare bitul citit în RAM1
avans adresă la RAM1
este adresa 920?
NU
da
-pregătire casetă pt oprire
- oprire fizică casetă
START
-pornire fizică casetă
- așteptare viteză nominală
- poziționare semnale scriere
- așteptare răspuns casetă
"pregătit scriere"
citire 1 bit din RAM2
scrie bitul pe casetă
avans adresă la RAM2
este adresa 920?
NU
DA
REDARE

Fig. 8.34.


Unitatea de comandă în funcțiune
C-AN
SYN (start)
INREGISTRARE
SYN(porn) 920 720
SYN (porn) bi porn
C-A/N
ACON

partea de sus a fig.3.33 unde segmentele verticale - ca lungime și ca poziție, reprezintă aproximativ momentul cînd este declanșat un dispozitiv de comandă și intervalul de timp cît acesta este în funcțiune.

La apariția semnalului SYN(start), este declanșată UC-I care determină selecția parametru cu parametru, asigurînd temporizarea de 12,5 ms între parametri, declanșează C-A/N cu semnalul ican, așteaptă sfîrșitul conversiei, acan, și formează SYN(bacan) Cu aceasta declanșează BC-H, iar cu semnalul pa-rt declanșează UC-I.

UC-H asigură toate comenzile necesare pentru efectuarea corecției numerice de scară și a conversiei binar - BCD, după care intră din nou în starea de așteptare a semnalului SYN(bacan),fără a anunța faptul că și-a încheiat misiunea.

UC-I, asigură încărcarea paralel a RI și descărcarea lui serie în RAM_1, precum și toate operațiile anexă (avans adresă, numărare biți, etc.).

In mod indirect UC-I asigură la fiecare grup de 720 de biți încărcați de RAM_1 generarea semnalului i 720_{10}^{1}(RAM_1-plin) care declanșează UC-H. Acesta copiază (RAM_1) în RAM_2, generează comanda "scr-cas" și intră în bucla de așteptare.

La comanda "scr-cas" se declanșează interfața casetei pe regim de înregistrare CAS-I, care este formată dintr-un automat secvențial sincron cu același nume, și un automat subordonat CAS-S care se ocupă efectiv de operația de scriere. Cuplajul dintre aceste două automate este de alt tip și anume:

- automatul principal declanșează automatul secundar și intră în regim de așteptare;

- automatul secundar (subordonat) își efectuează sarcinile, se aduce singur în stare de așteptare, și simultan predă comanda automatului principal care era în așteptare.

Deci la înregistrare, există în total în funcțiune la un moment dat 5 dispozitive de comandă (I,H,T,b,CAS-I sau CAS-S).

La redare, care este ilustrată în partea de jos a fig.1C, DC-P așteaptă semnalul SYN(porn) unde "porn" este comanda dată de un buton de pe PC.

Acest DC asigură distribuția parametru cu parametru la cele 8 ieșiri analogice ale SIR-RM, asigurînd și temporizarea de 12,5 ms.

La primul grup de selecții, de la declanșarea redării, $RAM_{1,2}$

nu conţin date şi pentru aceasta trebuiesc încărcate cu date de pe casetă; deci dă comanda "cit-cas" deoarece $b_1$porn=1.La prima trecere prin UC-T se aduce la "0" $b_1$porn şi în loc de "cit-cas" de acum încolo, va apare "cit" (pasul P2). In plus tot aici se aşteaptă "dval" - date valide. Pentru a nu complica figura, ieşirea "cit" din UC-P este notată cu A şi se duce la intrarea A a lui UC-T.

Deci să luăm cazul că apare "cit-cas", acesta declanşează automatul CAS-R, care se ocupă de dirijarea casetei în regimul de citire. Acesta comandă un automat subordonat CAS-C care se ocupă numai de citirea efectivă. In acest regim, interfaţa casetei, umple RAM_1 cu 720 biţi şi prin semnalul SYN(porn). $720^1_{10}\cdot 720^2_{10}$ se declanşează UC-M care face copierea din RAM_1 în RAM_2. In această UC se poate da comanda cit dacă $b_1$porn=1 prin care se declanşează UC-I, a cărui funcţionare este cunoscută, şi la care, la fiecare sfîrşit al ciclului de ceas se aduce la 0, $b_1$porn; datorită acestui fapt se declanşează UC-I, UC-H pentru a efectua de fapt numai procesele de selecţie - afişare în zecimal, a parametrilor de ieşire.

UC-T, extrăgînd mereu date din RAM_2, face să fie în final îndeplinită condiţia de declanşare a UC-M (SYN(porn).$720^1_{10}\cdot$ $\cdot 720^2_{10}$=1) - notaţiile C.

In toate celelalte cicluri UC-M generează "cit-cas" în loc de "cit", şi apare conexiunea B - B.

După cum se observă funcţionarea de ansamblu este foarte complexă şi a pus probleme deosebite, de cercetare şi proiectare necesitînd studii şi analize aprofundate.

9. CONCLUZII

În studiul, proiectarea și realizarea oricărui sistem numeric (SN), un rol esențial îl are cunoașterea structurii sale de comandă, cea mai complexă secțiune a sa, indiferent de faptul că SN poate fi de mică amploare (o interfață, un bloc numeric independent) sau amploare mare sau foarte mare (un calculator numeric, un sistem de calculatoare).

Stadiul actual al domeniului calculatoarelor, precum și previziunile pentru viitor conțin cele mai remarcabile realizări sau posibile realizări, nu în cadrul structurilor de date ci în cadrul structurilor de comandă. În cadrul structurilor de date modificările ce se prevăd sînt în marea lor majoritate de ordin cantitativ, pe cînd în cadrul structurilor de comandă se prevăd în special modificări calitative.

Noile noțiuni: migrarea funcțiilor, divergența comenzii, procesarea paralelă/întrețesută, anticiparea datelor (ca și a instrucțiunilor) au impact, în special cu structurile de comandă și mai puțin cu cele de date.

În prezenta teză de doctorat sînt analizate critic tipurile funcționalo-constructive prezente de structuri de comandă; sînt propuse noi structuri și noi metode de proiectare pentru structurile de comandă, sînt prezentate modele originale de sisteme divergente/convergente - la nivelul comenzii precum și unele realizări hardware originale organizate pe mai multe nivele ierarhice, cu execuții paralele, cu ierarhie flexibilă, etc.

Contribuțiile autorului sînt următoarele:

1. Clasificarea și analiza critică a tuturor tipurilor cunoscute de structuri de comandă de tip cablat, microprogramat, programat, combinat. Sînt analizate în detaliu, toate tipurile de secvențieri posibile: liniară, ciclică, ramificată, buclată, divergentă, convergentă, etc.

2. Separarea completă a structurii de comandă față de structura de date în cadrul unui SN, în vederea unificării metodelor de proiectare, în așa fel încît o structură de comandă care are perfect definite:

- intrările
- ieșirile

- funcţia de secvenţiere

să poată fi proiectată automat, fără a cunoaşte în detaliu structura de date aferentă.

3. Studiul comparativ, la nivel funcţional-constructiv al tehnicilor de proiectare şi al performanţelor, referitor la dispozitive de comandă (DC) sincrone/asincrone, cablate/microprogramate/programate.

Studiul este extins şi asupra DC realizate cu module PLA, ROM, MUX, DMUX, etc.

4. Prezentarea unor structuri existente sau posibile cu ierarhizarea comenzii, pe mai multe nivele, la care dialogul între nivele se poate face cu sau fără respectarea ierarhiei. Este prezentată o nouă structură de comandă în reţea, cu ierarhie flexibilă, cu interschimbabilitate "principal/subordonat" ("master"/"slave"). Se detaliază un model de dialog posibil între două componente ale unei astfel de structuri, în care orice componentă poate efectua simultan două operaţiuni (în paralel) şi anume: dialog-stare, dialog-date.

5. Analiza comparativă a tehnicilor şi metodelor de proiectare a structurilor de comandă, şi punerea în evidenţă a avantajelor/dezavantajelor acestor tehnici şi metode, precum şi definirea domeniilor de aplicabilitate.

6. Propunerea unui criteriu de alegere a tipului unui DC în funcţie de:

- viteza de operare
- preţul componentelor
- flexibilitatea cerută pentru DC
- preţul cablajelor
- simplitatea proiectării
- fiabilitatea impusă
- testabilitate, etc.

Acest criteriu de alegere este materializat printr-o nomogramă, a cărei tehnică de utilizare este prezentată în detaliu

7. Elaborarea unei metode de proiectare a dispozitivelor de comandă, care pleacă de la orice tip de descriere a protocolului de operare (organigramă, tabel, etc.). Metoda se pretează la proiectarea automată, datorită concisiunii şi preciziei ei, precum şi datorită faptului că nu conţine elemente descriptive sau de altă natură, neformalizate.

8. Prezentarea unei organigrame şi a unui program, referitoare la metoda de mai sus. Cu ajutorul programului s-a verificat pe exemple concrete, valabilitatea metodei elaborate (v.anexa 1).

9. Adaptarea unor metode de proiectare a structurilor de comandă la proiectarea unor structuri de date cu funcţionare în secvenţă. In acest fel, s-a proiectat o celulă de convergenţă pentru două ramuri ale unui DC cu execuţii paralele, atît în varianta "automat cu stări codificate" cît şi în varianta "automat cu stări necodificate".

10. Delimitarea domeniilor de aplicabilitate a structurilor seriale/piramidale pentru reţele de celule elementare de convergenţă, în cazul necesităţii unei convergenţe multiple.

11. Prezentarea unei noi soluţii pentru convergenţa multiplă, bazată pe utilizarea unui registru de convergenţă.

12. Propunerea unei tehnici de segmentare a DC de mare complexitate, similară tehnicilor de segmentare a programelor. Evidenţierea condiţiilor şi locurilor pentru care segmentarea este posibilă.

13. Analiza comparativă a tehnicilor de cuplare a mai multor DC. Elaborarea unei tehnici noi de segmentare şi cuplare a dispozitivelor de comandă care înlătură posibilitatea apariţiei hazardului în funcţionare şi elimină necesitatea analizei corelării perioadelor de tact a diverselor automate; de asemenea s-au simplificat şi modularizat elementele implicate în cuplare.

14. Prezentarea unor posibilităţi de reducere a timpilor morţi la cuplarea unor automate prin modificări minore ale protocolului de operare în zonele afectate de cuplare.

15. Propunerea unei structuri de comandă pentru un calculator care pe lîngă posibilitatea de aducere anticipată a instrucţiunilor are implementată şi posibilitatea de aducere anticipată a operanzilor. Structura propusă, este de tipul "cu funcţionare în paralel" şi conţine şi un modul cumulativ, de unificare a comenzilor.

16. Pentru cazurile mai complexe de procesări sau execuţii paralele în care poate exista conflict de cuplare la ni-

velul mai multor DC sau la nivelul altor resurse hardware implicate în cuplare s-a propus un alt tip de structură bazată pe registre de cereri de cuplare, registre de stare a sistemului de cuplare, arbitru de priorităţi la cuplare,etc.

17. Prezentarea unei structuri de comandă originale, utilizată în cadrul unui stand de testare a memoriilor,formată dintr-un sistem de comandă ierarhizat pe cinci nivele, supervizată de un sistem monoprocesor.

In cadrul acestei structuri s-au aplicat şi demonstrat practic,principiile enunţate în capitolele referitoare la segmentarea, cuplarea şi ierarhizarea structurilor de comandă.

18. Prezentarea unei structuri de comandă originale, cu execuţii paralele, cu mai multe DC "sleve", supervizate de două DC "master" şi ierarhizarea variabilă (flexibilă), utilizată la achiziţie/distribuţie de date în timp real.

19. In cadrul structurii de la punctul 18 este descris şi proiectat un automat capabil ca pentru sistemele de achiziţie de date să asigure efectuarea corecţiei de scară şi a conversiei de cod pentru mărimile fizice achiziţionate.

Sistemele de comandă realizate ca parte experimentală a acestei lucrări, demonstrează corectitudinea principiilor enunţate, referitoare la segmentare şi cuplare, ierarhizarea comenzii, paralelisme în cadrul protocoluţilor de operare, precum şi cele referitoare la posibilitatea existenţei unei ierarhii flexibile. Astfel de principii pot fi utilizate în cadrul noilor arhitecturi ale SN din generaţia a 5-a.

Noua metodă de proiectare a dispozitivelor de comandă, asistată de calculator, are perspectiva să devină aplicabilă la proiectarea unor sisteme de calcul cu arhitecturi ce nu respectă principiile lui Von Neumann.

Lucrările din prezenta teză de doctorat şi-au găsit aplicabilitate în cadrul unor contracte de cercetare pentru I.T.C. -filiala Timişoara, Intreprinderea "Electrobanat" şi "Institutul pentru sudură şi încercări de materiale",Timişoara.

Studiile sînt continuate de autor, în acest domeniu, întrezărindu-se o perspectivă nouă pentru sistemele numerice din generaţia a 5-a.

10. BIBLIOGRAFIE

1. A1. ABRAMSON, I.B., KUO, F.F.,-Computer Communications Networks, Prentice Hall, Englewood Cliffs, N.J.,1973.

2. A2. AGRAWALA, A.K., BAUSHER, T.G., Microprogramming Perspective and Status, IEEETC, vol.C-23, p.817-837,Aug. 1974.

3. A3. ASHAR, S.B., ROUTING (cap.6) in M.A.BREUER, Design Automation of Digital Systems, Prentice Hall, Englewood, N.J., 1972.

4. A4. ANDI, L., Field Programmable Logic Array, "Progress" Fairchild journal of semiconductor, vol.6,nr.4,iul.-aug.1976.

5. B1. BALLAC, V., s.a., Sisteme interactive, limbaje conversaționale, Ed.Pol.Bucureşti, 1984.

6. B2. BALLAC, V., s.a., Calculatoarele electronice, grafică interactivă şi prelucrarea imaginilor, Ed.Tehn. Buc.1985.

7. B3. BARBACCI, M.R., A Comparison of Register Transfer Languages for Describing Computers and Digital Systems, IEEETC, vol.C-24,feb.1975.

8. B4. BEELA, A., PORAT, I.D., Integrated circuits in digital electronics, New Jork, John Willey and Sons Inc. 1973.

9. B5. BARNES, G.H.,s.a., The ILLIAC IV Computer, IEEETC, aug.1968, p.746.

10. B6. BASU, I.S., NAWAL, A., Computer Structures:Readings and Examples, Mc Graw Hill, 1971.

11. B7. BLAAUW, G.A., Digital System Implementation, Prentice Hall, Englewood Cliffs, N.J., 1976.

12. B8. BRAUN, E.L., Digital Computer Design, Academic Press, N.J., 1973.

13. B9. BREUER, M.A., A Random algorithm Techniques for Fault Detection Test Generation for Sequential Circuits, IEEETC, vol.C-20, 1971.

14. B10. BREUER, M.A., FRIEDMAN, A.D., Diagnosis and Reliable Design of Digital Systems, Computer Science Press, Inc. Woodland Hills, 1976.

15. C1. CARR, W.T., MIZE, J.P., MOS/LSI Design and application New Jork, Mc Graw Hill Book Co., 1972.

16. C2. CASSAGLIA, G.F., OLIVETTI, I.C., Nonprogramming vs. Microprogramming, Computer, Jan. 1976, p.54-56.

17. C3. CAVLAN, N., Field Programmable Device, "Signetics"-Design manual.

18. C4. CAPATINA, O.D., HASEGAN, M.C., PUSCA, M.V., Proiectarea cu microprocesoare, Ed.Dacia, Cluj-Napoca, 1983.

19. C5. CHECHANU, M.D., Construcţia calculatoarelor, Institutul Politehnic "Traian Vuia" Timişoara, 1985.

20. C6. CHECHANU, M.D., NITU, A., BERECZKY, F., BALEA, L., Testor pentru module de memorie MOS, IPTVT, a XX-a Ses.de com.şt.stud., 19-20 mai 1979.

21. C7. CHECHANU, M.D., STRATULAT, M., MIHAILESCU, A., Prelucrarea analog-numerică a parametrilor sudării automate, 4 th International Conference on Control Syetems and Computer Science, Bucharest 17-20 June,1981.

22. C8. CHECHANU, M.D., Unitate de comandă pentru testul Galloping la o memorie MOS, Referat Doctorat,1981.

23. C9. CHECHANU, M.D., Sistem de achiziţie, înregistrare şi distribuţie a parametrilor sudării automate, Referat Doctorat, 1982.

24. C10. CHECHANU,M.D., STRATULAT, M., MIHAILESCU, A., MOS,I., Sistem de înregistrare şi reproducere sub formă numerică a mărimilor analogice, CNETAC, Buc.,1982.

25. C11. CHECHANU,M.D., STRATULAT, M., MIHAILESCU, A., MOS,I., Sistem numeric pentru conducerea unor procese de sudare, CNETAC, Buc., 1984.

26. C12. CHECHANU, M.D., STRATULAT, M., MIHAILESCU, A., Prelucrarea analog-numerică a parametrilor sudării automate, A 4-a conferinţă de sisteme automate şi informaţionale în industrie, mai 1981,Buc.

27. C13. CHECHANU, M.D., STRATULAT,M., MIHAILESCU,A.,AOS,I. C.,Sistem de înregistrare şi redare a unor mărimi analogice avînd ca suport de informaţie-banda magnetică. Certificat de inovator, 18.06.1985,IPTVT.

28. C14. CHECHANU,M.D.,STRATULAT,M.,MIHAILESCU,A., Prelucrarea "analog-numerică a parametrilor de sudare automată. Colocviul de cibernetică" organizat de Acad. RSR, baza de cercetări ştiinţifice Timişoara,nov. 1981.

29. C15. CHECHANU, M.D., Asupra corecţiei de scară şi a conversiei de cod în sistemele de achiziţie de date, SAMIAC,Buc.1986.

30. C16. CHECHANU, M.D., Utilizarea unui cuplu 2xRAM-FIFO în sistemele de achiziţie-înregistrare-distribuţie, SAMIAC, Buc.1986.

31. C17. CHECHANU, M.D., POPESCU, T., Aparat pentru testarea dispozitivelor numerice, Sesiunea comună de comunicări a cadrelor didactice şi studenţilor,IPTVT,1978

32. C18. CLARK,C.R., Designing logic systems using state machines, Mc Graw-Hill Book Company,1972.

33. C19. CHEN, T.C., Parallelism, Pipelining, and Computer Efficiency, Computer Design, ian.1971,p.69.

34. C20. CHU, Y., WHY DO WE NEED HARDWARE DESCRIPTION LANGUAGES ? , Computer, dec.1974.

35. C21. CHU, Y., Computer Organization and Microprogramming Prentice Hall, New Jersey, 1972.

36. C22. CHU, Y., Bazele proiectării calculatoarelor numerice, Ed.Tehnică, Buc.,1968.

37. C23. COSMA, O.,g.a.,Proiectarea asistată de calculator a sistemelor discrete, Ed.Acad.RSR,Buc.,1984.

38. D1. DANCEA, I., Microprocesoare, Arhitectură internă programare, aplicații. Ed.Dacia Cluj-Napoca,1979.

39. D2. DAVIDOVICIU, A.,g.a., Minicalculatoarele și microcalculatoarele în conducerea proceselor industriale E.T.Buc., 1985, Ed.Maginostroenie Leningrad,1984.

40. D3. DIETMEYER, D.L. Logical Design of Digital Systems Allyn and Bacon, Boston, 1971.

41. D4. DAVIS, C.C., g.a. Gate array embodies System/370 processor, Electronics, oct.9,p.140-145,1980.

42. D5. DODESCU,G.,g.a. Minicalculatoare-aplicații,Ed.Tehn. București, 1978.

43. D6. DRĂGĂNESCU,M., A doua revoluție industrială,Microelectronica, Automatica,Informatica-factorii determinanți, Ed.Tehnică, Buc.1980.

44. D7. DULEY,J.R., DIETMEYER,D.L.,A Digital System Design Language, IEEETC,C-17,sept.1968.

45. E1. ENSLOW,P.H., Multiprocessors and Parallel Processing (ediția în limba rusă),Ed.Mir,Moscova,1976.

46. E1. EREIGENBAUM,E., Mc.CORDUCK,P., La cinquieme génération, Inter Editions, Paris,1984.

47. F2. FLYNN,M.J., Some Computer Organizations and Their Effectiveness, IEEETC,vol.C-21,p.948-960.

48. F3. FLORES,I., Computer Organization Prentice-Hall,Englewood Cliffs, N.J., 1969.

49. F4. FRIEDMAN, T.D., YANG,S.C., Methods Used in an Automatic Logic Design Generator (ALERT),IEEETC,vol.C-18, July,1969.

50. G1. GERACE,G.B., Microprogramm Control for Computing systems, I.R.E.Trans Elec.Computer,vol.EC-12,1963.

51. G2. GERER,T.,g.a. Echipamente periferice, Ed.Tehn.Buc. 1981.

52. H1. HARLOW,C.A., COATES,C.L., Feedback in Sequential Machine Realizations, IEEETC,apr.1972,vol.C-21,nr.4, p.371-381.

53. H2. HASTERLIK,R.L., RTL-The firmware design automation system, Design Automation Workshop,1974.

54. H4. HELLERMANN,G., CONROY,T.F., Computer System Performance, McGraw-Hill,New Jork,1975.

55. H5. HANGANUT,M., Utilizarea calculatoarelor in procese industriale, Cluj-Napoca.

56. H6. HILL,F.J., Introducing AHPL, Computer,dec.1974,p.26 30.

57. H7. HILL,F.J., PETERSON,G.R., Calculatoare numerice Harware -structură şi proiectare,Ed.Tehn.Buc.1980.

58. H8. HUSSON,S.S., Microprogramming, Principles and Practice, Prentice-Hall, Inc.New Jersey, 1970.

59. I1. IBBARA,O.H., On the Equivalence of Finite-State Sequential Machine Models, IEEETC,vol.EC-16 nr.1,feb. 1967,p.88-90.

60. I2. IONESCU,T., Sisteme şi echipamente pentru conducerea proceselor, Ed.Did.şi Ped.,Buc.1982.

61. I3. IVERSON,K.E., A Common for Harware,Software and Applications, Proc.Fall Joint,Comput.Conf.1962.

62. J1. JONES,T., THOMAS,P., Challenges in Microprocessor System Design, Computer Design,nov.1976.

63. K1. KORNSTEIN,H., High Performance Microprocessor Structure, Systems, vol.7,nr.5,may 1979,p.39-41.

64. K2. KOVACS,F., COJOCARU,G., Manipulatoare, roboti şi aplicaţiile lor industriale, Ed.Facla,Timişoara,1982.

65. K3. KLINE,R.M., Digital Computer Design, Prentice Hall, Englewood Cliffs,N.J.,1977.

66. K4. KLIR,J., A Note on the Basic Block Diagrams of Finite Automata from the Enginnering Point of View IEEETC,vol.EC-16,nr.2,apr.67,p.223-224.

67. L1. LESEA,A., ZAKS,R., Microprocessor Interfacing Techniques, Berkeley/Ca, Sybex Publication,1977.

68. L2. LEWIN,D., Theory and Design of Digital Computers, London, Nelson and Sons.Ltd.,1972.

69. L3. LIPORSKY,G.J., Hardware Description Languages, Computer, June, 1977.

70. M1. MANO, M., Computer Logic Design, Englewood Cliffs, Prentice-Hall, Inc.,1972.

71. M2. MARTIN,D.P., Microcomputer Design, ed.2., Martin Research, Northbrook, 1976.

72. M3. Mc GLYNN, D., Microprocessors, Technology Architecture and Applications, New Jork, John Willey and Sons, Inc.,1976.

73. M4. MEALY,G., A Method forS Syntetizing Sequential Circuits, Bell System Technical Journal, 34,1045-1079, 1955.

74. M5. MUNTEAN,I., Sinteza automatelor finite, Ed.Tehn.Buc. 1977, Seria Matematici Moderne Aplicate.

75. M6. MURESAN, T., STRUGARU,C.,g.a. Microprocesorul 8080 in aplicatii, Editura Facla Timisoara,1981.

76. N1. NEMEC, J., A Primer on Bit-Slice Processors,Electronic Design , febr.1977.

77. N2. NICOLAU, E., POPOVICI,AL., Introducere in cibernetica sistemelor hibride, Ed.Tehn.Buc., 1975.

78. O1. ORBORNE,A., An Introduction to Microcomputers,Vol.1, Basic Concepts. Berkeley Ca.,Sybex Publ.,1976.

79. O2. ORBORNE,A., 8080 Programming for Logic Design, Berkeley Ca., Sybex Publication, 1976.

80. P1. PAULL,M.C., WALDHAUM,C., A note on State Minimisation of Asyncronous Sequential Functions IEEETC,vol.EC-16, nr.1, feb.1967,p.94-97.

81. P2. PEATMAN, I.B., Microcomputer-Based Design, Mc Graw-Hill, N.Y.,1977.

82. P3. PETRESCU,A., Sisteme sistolice de prelucrare a datelor in Calculatoarele electronice din generatia a 5-a, Ed.Acad.RSR,Buc.,1985.

83. P4. PETRESCU,A.,g.a.,Microcalculatoarele FELIX M18,M18B, M118, Ed.Tehn.Buc.1984.

84. P5. PATRESCU,M., Eficienţa integrării pe scară medie şi largă asupra tehnicilor de sinteză a circuitelor combinaţionale şi secvenţiale, IPB, 1979.

85. P6. PATRESCU, M., Dezvoltarea ştiinţei sistemelor cibernetice în România în "Istoria ştiinţelor în România, Cibernetica," Ed.Academiei RSR, Buc.,1981.

86. P7. PLUMMER, W.W., Asynchronous Arbiters, IEEETC,vol.C21 nr.1, ian.1972,p.37-42.

87. P8. POP, V., Structura sistemelor de prelucrare a datelor numerice, IPTVT, 1981.

88. P9. POP, V., Bazele logice ale calculatoarelor, IPTVT, 1974.

89. R1. READ, I.S., Symbolic Synthesis of Digital Computers Proc.ACM, Toronto, Sept.,1952.

90. R2. ROGOJAN, AL., POP, V., STRUGARU, C., STRATULAT, M., CHECEANU, M.D., ş.a. Testor de baterii electrice, BSTIPTVT, Tom,25(39), iul-dec.1980, fasc.2.

91. R3. ROGOJAN, AL., STRUGARU, C., CHECEANU M.D.,ş.a. Testor pentru amplificatoare de citire, BSTIPTVT,Tom 25(39), iul.-dec., 1980, fasc.2.

92. R4. ROGOJAN, AL., CHECEANU, M.D.,ş.a. Sistem de achiziţii de date, BSTIPTVT, Tom 27(41), ian.-iun.,1982, fasc.1.

93. R5. ROGOJAN, AL., Calculatoare numerice, IPTVT, 1975.

94. R6. ROGOJAN, AL., POP, V., STRUGARU,C., CHECEANU, M.D., ş.a., Testor de baterii condus cu calculator de proces. Simpozionul "Domenii de utilizare a calculatoarelor electronice în ştiinţă şi tehnică, org. de Acad.RSR, Baza de cercet.Timişoara, mai,1981.

95. R7. ROTH, J.P., Diagnosis of Automata Failures:A Calculus and a Method, IBM Journal, 10, 1966.

96. S1. SAUCIER, G., Next-State Equations of Asyncronous Sequentical Machines, IEEETC, apr.1972,p.397-399.

97. S2. SCHOOR, M., Computer Aided Digital System, Design and Analysis Using a Register Transfer Language, IEEETC, dec.1964, p.730-737.

98. S3. SLOAN, M.E., Computer Hardware and Organization Science Research Associates, CHICAGO, 1976.

99. S4. SOUCEK, B., Microprocessors and Microcomputers, New York, John Willey and Sons, Inc., 1976.

100. S5. STRATULAT, M., CHICHEANU, N.D., MIHAILESCU, A., Inregistrarea si redarea parametrilor de sudare in instalatii automate de sudura. Ses.de com.st.ocazionata de implinirea 20 ani de existenta a Institutului de Invatamint Superior din Constanta, oct.1981

101. S6. STRINCEANA, N., DOBRESCU, R., BOHANGIU, T., Automatizari discrete in industrie-culegere de probleme, Ed.Tehn.Buc.1978.

102. S7. SAMPATEANU, M., Circuite pentru conversia datelor, Ed.Tehn.Buc.1980.

103. S8. STRUGARU, C., CHICHEANU, N.D., s.a. Stand pentru incercarea de durata in vederea determinarii fiabilitatii modulelor de memorie, Buletinul Stiintific si Tehnic al IPTVT, Tom.26(40), ian.-iun., 1981, Fasc.1.

104. T1. TANNENBAUM, A.S., Structured Computer Organization, Prentice Hall, Englewood Cliffs, N.J., 1976.

105. T2. TEODORESCU, D., Introducere in microelectronica, Editura Facla, Timisoara, 1985.

106. T3. TOACSE, G., Introducere in microprocesoare, Editura Stiintifica si Enciclopedica, Bucuresti, 1985.

107. T4. TORNG., C.N., WILHELM, N.C., The Optimal Interconnection of Circuit Modules in Microprocessor and Digital System Design, IEEETC, C-26, may, 1977.

108. T5. TOTH, A., HOLT, C., Automated Data-Base-Driven, Digital Testing, Computer, 1, 1974.

109. T6. THURBER, K.J., Parallel Processors Architectures Computer Design, vol.18, nr.1, ian.1979, p.89-97 si nr.2, febr.1979, p.103-14.

110. W1. WEINER,P.,SMITH, E.J., On the Number of Distinct State Assignments for Synchronous Sequential Machines, IEEETC,Vol.EC-16,nr.2,apr.1967.

111. W2. WEINER,P.,SMITH,E.J.,Optimization of Reduced Dependencies for Synchronous Sequential Machines, IEEETC,vol.EC-16,nr.6,dec.1967,p.835-847.

112. W3. WILLES,A.V.,Sisteme de calcul cu acces multiplu, Editura Tehn.,Buc.,1979.

113. Z1. ZAKS,R., Microprocessors From Chips to Systems, Berkeley, Ca., Sybex Publication, 1977.

114. *1. *** Electronics. Designer's Casebook. A Mc Graw-Hill Publication.

115. *2. *** Smal computer handbook Digital Equipment Corporation, 1970 Edition.

116. *3. *** Standard Microsystems Corporation,Data Catalog, 1980.

117. *4. *** Calculatoarele electronice din generatia a 5-a, Edit.Acad.RSR,Buc.,1985.

118. *5. *** National Semiconductor, 4096 bit dinamic RAM program, 1977.

119. *6. *** Guide to Microcomputer Development Systems, INTEL.

120. *7. *** The 68000 Harware and Software

121. *8. *** The Integrated Circuits Catalog for Design Engineers, Texas, Instr.Inc.

122. *9. *** The TTL Data Book, Supplement to CC-401 for Design Engineers Texas Instruments,Inc.

Obs. In lista bibliografică s-au utilizat prescurtările:
-BSTIPIVT:Buletinul Stiintific şi Tehnic al Institutului Politehnic "Traian Vuia",Timişoara.
-IEEETC: IEEE Transactions on Computers.

ANEXA 1

```
DIM C%(9%,5%),R%(9%,5%),F%(9%,5%)
DIM D1$(10%),D2$(9%),D3$(9%)
K1%=10%
K2%=5%
K3%=9%
K4%=4%
MAT C%=ZER
MAT R%=ZER
MAT F%=ZER
FOR I%=1% TO K3%
D1$(I%)=' '
D2$(I%)=' '
D3$(I%)=' '
NEXT I%
D1$(I%)=' '
PRINT ' '
PRINT ' Introducere stari : '
PRINT ' '
I%=1%
INPUT ' Stare ',V$
IF V$ <> '-1' GOTO 210
N1%=I%-1%
GOTO 260
D1$(I%)=V$
I%=I%+1%
IF I% <= K1% GOTO 175
PRINT ' * Depasire nr stari *' :
GOTO 1510
N2%=0%
N3%=0%
PRINT ' '
I%=1%
PRINT ' '
J%=1%
PRINT ' Starea ';D1$(I%),' conditia ';J%
INPUT ' Conditie ? ',V$
IF V$ <> '-1' GOTO 380
C%(I%,J%)=-1%
I%=I%+1%
IF I% <= K1% GOTO 290
PRINT ' * Depasire nr stare *'
GOTO 1510
IF V$ = '-2' GOTO 785
IF V$ = ' ' GOTO 500
K%=1%
IF V$ = D2$(K%) GOTO 490
IF D2$(K%) = ' ' GOTO 470
K%=K%+1%
IF K% <= K3% GOTO 410'
PRINT ' * Depasire dictionar conditii *'
GOTO 1510
D2$(K%)=V$
N2%=N2%+1%
C%(I%,J%)=K%
INPUT ' Ramificatie ! ',V$
IF V$ = ' ' GOTO 630
K%=1%
IF V$ = D1$(K%) GOTO 620
IF D1$(K%) = ' ' GOTO 590
```

```
550     K%=K%+1%
560     IF K% <= K3% GOTO 530
570     PRINT ' * Depasire dictionar stari *'
580     GOTO 1510
590     PRINT ' ' Secventa absenta ! ';V$
600     D1$(K%)=V$
610     N1%=N1%+1%
670     R%(I%,J%)=K%
630     INPUT ' Functie ! ',V$
640     IF V$ = ' ' GOTO 750
650     K%=1%
660     IF V$ = D3$(K%) GOTO 740
670     IF D3$(K%) = ' ' GOTO 720
680     K%=K%+1%
690     IF K% <= K3% GOTO 660
700     PRINT ' * Depasire dictionar functii *'
710     GOTO 1510
720     D3$(K%)=V$
730     N3%=N3%+1%
740     F%(I%,J%)=K%
750     J%=J%+1%
760     IF J% <= K2% GOTO 300
770     PRINT ' * Depasire nr conditii *'
780     GOTO 1510
785     PRINT ' '
790     PRINT ' Sfirsit culegere date '
795     GOSUB 1260
800     I1%=1%
810     I%=1%
820     M$='  D:'+D1$(I1%)+'='
830     S%=0%
840     J%=1%
850     IF C%(I%,J%) <> A1% GOTO 910
860     I%=I%+1%
870     IF I% <= N1% GOTO 840
880     PRINT M$
890     I1%=I1%+1%
```

```
900     IF I1% <= N1% GOTO 810
905     GOTO 1020
910     IF R%(I%,J%) <> I1% GOTO 1000
920     IF S% = 0% GOTO 950
930     M$=M$+'+'
940     GOTO 960
950     S%=1%
960     M$=M$+D1$(I%)
970     IF C%(I%,J%) = 0% GOTO 1000
980     K%=C%(I%,J%)
990     M$=M$+'*'+D2$(K%)
1000    J%=J%+1%
1010    GOTO 850
1020    PRINT
1030    I3%=1%
1040    I%=1%
1050    M$='  '+D3$(I3%)+'='
1060    S%=0%
1070    J%=1%
1080    IF C%(I%,J%) <> -1 GOTO 1150
1090    I%=I%+1%
1100    IF I% <= N1% GOTO 1070
1110    PRINT M$
1120    I3%=I3%+1%
1130    IF I3% <= N3% GOTO 1040
1140    GOTO 1510
1150    IF F%(I%,J%) <> I3% GOTO 1240
1160    IF S% = 0% GOTO 1190
1170    M$=M$+'+'
1180    GOTO 1200
1190    S%=1%
1200    M$=M$+D1$(I%)
1210    IF C%(I%,J%) = 0% GOTO 1240
1220    K%=C%(I%,J%)
1230    M$=M$+'*'+D2$(K%)
```

```
1240    J%=J%+1%
1250    GOTO 1080
1260    PRINT ' ' FOR I%=1% TO 5%
1270    PRINT ' Tabel intrari :'
1280    PRINT ' '
1290    FOR I%=1% TO N1%
1300    PRINT D1$(I%)
1310    FOR J%=1% UNTIL C%(I%,J%)=-1%
1320    PRINT '    ';C%(I%,J%);R%(I%,J%);F%(I%,J%)
1330    NEXT J%
1340    NEXT I%
1350    PRINT ' ' FOR I%=1% TO 5%
1360    PRINT ' Dictionar stari :'
1370    PRINT ' '
1380    PRINT ' ';D1$(I%),I% FOR I%=1% TO N1%
1390    PRINT ' ' FOR I%=1% TO 3%
1400    PRINT ' Dictionar conditii :'
1410    PRINT ' '
1420    PRINT ' ';D2$(I%),I% FOR I%=1% TO N2%
1430    PRINT ' ' FOR I%=1% TO 3%
1440    PRINT ' Dictionar functii  :'
1450    PRINT ' '
1460    PRINT ' ';D3$(I%),I% FOR I%=1% TO N3%
1470    PRINT ' ' FOR I%=1% TO 5%
1480    PRINT ' Ecuati :'
 490    PRINT ' '
 500    RETURN
1510    END
;
!AZ


BASIC-PLUS - V4.1 TERMINATED AT 85/11/21  09:17:20
```

```
  READY  ***

N PROGR

troducere stari I

are ?A
are ?B
are ?C
are ?D
are ?E
are ?F1
are ?F2
are ?G
are ?-1

area A      conditia  1
nditie : ?a
nificatie : ?B
nctie : ?
area A      conditia  2
nditie : ?Na
ificatie : ?A
nctie : ?
area A      conditia  3
nditie : ?-1

area B      conditia  1
nditie : ?
mificatie : ?C
nctie : ?
area B      conditia  2
nditie : ?-1

area C      conditia  1
nditie : ?
mificatie : ?D
nctie : ?x
area C      conditia  2
nditie : ?-1

area D      conditia  1
nditie : ?b
ificatie : ?E
nctie : ?y
area D      conditia  2
nditie : ?Nb
ificatie : ?F1
nctie : ?
area D      conditia  3
nditie : ?Nb
ificatie : ?F2
nctie : ?
area D      conditia  4
nditie : ?-1

area E      conditia  1
nditie : #
ificatie : ?G
nctie : ?
area E      conditia  2
nditie : ?-1
```

```
Starea F1       conditia  1
Conditie : ?c*d
Ramificatie : ?
Functie : ?
Starea F1       conditia  2
Conditie : ?c*Nd
Ramificatie : ?G
Functie : ?
Starea F1       conditia  3
Conditie : ?Nc*d
Ramificatie : ?G
Functie : ?
Starea F1       conditia  4
Conditie : ?Nc*Nd
Ramificatie : ?A
Functie : ?
Functie : ?
Starea F1       conditia  5
Conditie : ?-1

Starea F2       conditia  1
Conditie : ?
Ramificatie : ?C
Functie : ?
Starea F2       conditia  2
Conditie : ?-1

Starea G        conditia  1
Conditie : ?
Ramificatie : ?B
Functie : ?
Starea G        conditia  2
Conditie : ?-1

Starea          conditia  1
Conditie : ?-2

Sfirsit culegere date
```

Tabel intrari :

```
A
     1   2   0
     2   1   0
B
     0   3   0
C
     0   4   1
D
     3   5   2
     4   6   0
     4   7   0
E
     0   8   0
F1
     5   0   0
     6   8   0
     7   8   0
     8   1   0
F2
     0   3   0
G
     0   2   0
```

ctionar stari :

1
2
3
4
5
6
7
8

ictionar conditii :

1
2
3
4
5
6
7
8

ctionar functii :

1
2

ati :

=C
=D*L
OF PROGRAM AT LINE 51)

ANEXA 2.

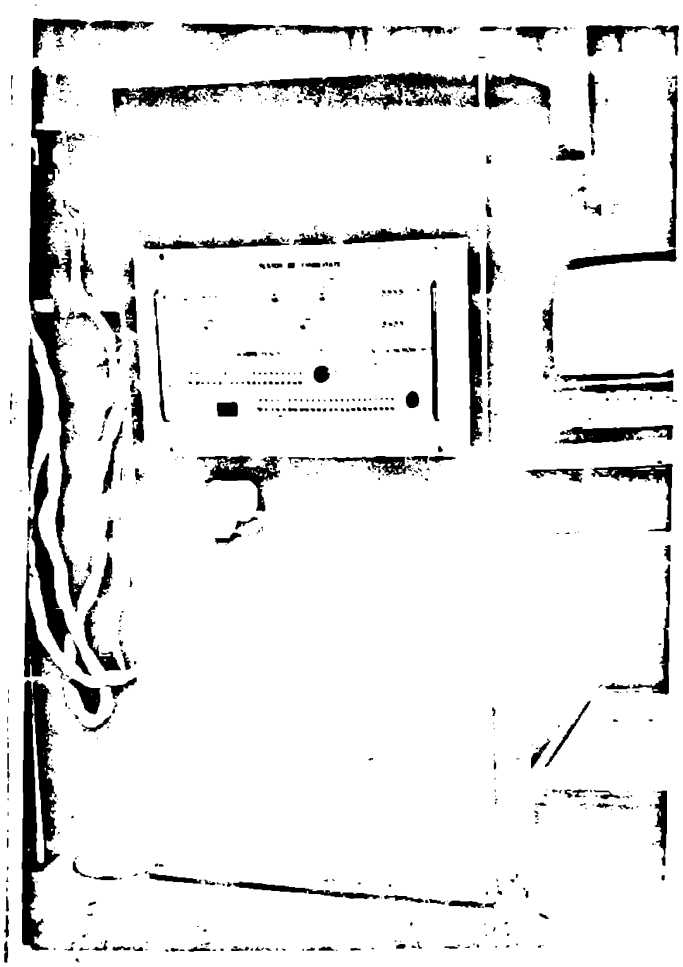


FOTO 1