INSTITUTUL POLITEHNIC "TRAIAN VUIA" TIMISOARA
FACULTATEA ELECTROTEHNICA

CIUGUDEAN MIRCEA

ALGORITMI SI STRUCTURI LOGICE CELULARE DE GENERARE A LOGARITMILOR SI ANTILOGARITMILOR BINARI PENTRU DISPOZITIVE ARITMETICE CU VIRGULA MOBILA

TEZA DE DOCTORAT

BIBLIOTECA CENTRALĂ
UNIVERSITATEA "POLITEHNICA"
TIMIŞOARA

CONDUCATOR STIINTIFIC
PROF.DR.ING.ALEXANDRU ROGOJAN

INSTITUTUL POLITEHNIC TIMIŞOARA
BIBLIOTECA CENTRALĂ
Volumul Nr. [illegible]
Dulap [illegible] Lit. [illegible]

BUPT

BUPT

C O N T I N U T

pag.

INTRODUCERE 3

CAPITOLUL I. ALGORITMI SI CIRCUITE CUNOSCUTE DE GENERARE A LOGARITMILOR SI ANTILOGARITMILOR BINARI . . . 16

1.1. Algoritmi de precizie redusă 16

1.1.1. Algoritmul Mitchell 16

1.1.2. Algoritmul Combet 19

1.1.3. Primul algoritm Dean 21

1.1.4. Algoritmul Hall 23

1.1.5. Algoritmul Marino 24

1.1.6. Algoritmul Julien 24

1.1.7. Algoritmul Andrews 25

1.1.8. Algoritmul Nicaud 26

1.2. Algoritmi de precizie ridicată 28

1.2.1. Al doilea algoritm Dean 28

1.2.2. Al treilea algoritm Dean 29

1.2.3. Algoritmul Perle 35

1.2.4. Algoritmul Schmid 38

CAPITOLUL II. NOI ALGORITMI PENTRU CALCULUL LOGARITMILOR SI ANTILOGARITMILOR BINARI 40

2.1. Algoritm bazat pe înmulţirea şi împărţirea cu constante a numărului ce se logaritmează sau antilogaritmează 40

2.2. Algoritm bazat pe descompunerea în factori a numărului ce se logaritmează 46

2.3. Calculul constantelor L_i şi C_i cu ajutorul calculatorului electronic 59

CAPITOLUL III. STRUCTURI LOGICE CELULARE PENTRU GENERAREA LOGARITMILOR SI ANTILOGARITMILOR BINARI . . . 66

3.1. Dispozitiv de generare a logaritmilor şi antilogaritmilor binari ce include o structură logică celulară de înmulţire şi împărţire 66

3.2. Structură logică celulară de generare în paralel a antilogaritmilor binari bazată pe înmulţirea constantelor C_i 74

3.3. Structuri logice celulare de generare a logaritmilor şi antilogaritmilor binari bazate pe descompunerea în factori şi termeni 78

BUPT

3.3.1. Structuri logice celulare pentru generarea logaritmilor binari . . . . pag. 79
3.3.2. Structuri logice celulare pentru generarea antilogaritmilor binari 93
3.3.3. Posibilităţi de comasare a structurilor logice celulare de generare a logaritmilor şi antilogaritmilor 1o2
3.3.4. Posibilităţi de îmbunătăţire a vitezei structurilor logice celulare de logaritmare şi antilogaritmare 1o8

CAPITOLUL IV. SIMULAREA PE CALCULATOR NUMERIC A STRUCTURILOR LOGICE CELULARE DE LOGARITMARE SI ANTILOGARITMARE 135
4.1. Program de simulare a unei scheme logice combinaţionale bazat pe urmărirea selectivă. 135
4.2. Simularea structurilor logice celulare de generare a logaritmilor şi antilogaritmilor binari 146

CAPITOLUL V. DISPOZITIV ARITMETIC CU VIRGULA MOBILA LOGARITMIC 151
5.1. Algoritmi pentru operaţiile cu logaritmi. 152
5.1.1. Algoritmul pentru înmulţire şi împărţire. 153
5.1.2. Algoritmul pentru ridicare la patrat şi extragerea rădăcinii patrate 155
5.1.3. Algoritmul pentru logaritmare 157
5.1.4. Algoritmul pentru antilogaritmare 158
5.2. Structura dispozitivului aritmetic cu virgulă mobilă logaritmic 159

CONCLUZII 168

BIBLIOGRAFIE 172

BUPT

INTRODUCERE

La marea majoritate a calculatoarelor electronice numerice universale există instrucţii "cablate" pentru cele patru operaţii aritmetice fundamentale : adunare, scădere, înmulţire şi împărţire iar operaţiile aritmetice mai complexe - ridicarea la putere, extragerea rădăcinii şi altele - se realizează prin calcule pe baza unor subprograme speciale.

Ideea generării logaritmilor cu ajutorul circuitelor electronice şi a utilizării lor în dispozitivele aritmetice [1] a apărut în mod firesc ca o transpunere în calculator a metodei de calcul cu logaritmi din matematică.

Destinaţia principală a logaritmilor obţinuţi prin hardware a fost aceea de înlocuire a operaţiilor relativ complicate de înmulţire şi împărţire printr-o simplă adunare sau scădere [1], [2], [6], [7], [25], [45], [62]. In calculatoarele electronice universale această înlocuire este posibilă însă numai dacă se admit rezultate fracţionare în calcule, dacă se realizează precizie suficient de mare în operaţiile de logaritmare şi antilogaritmare şi numai dacă se obţine un avantaj în cel puţin una din direcţiile : viteză de calcul şi costul circuitelor electronice, faţă de dispozitivele aritmetice obişnuite. In literatura de specialitate nu este amintit nici un caz de calculator numeric universal care să conţină circuite pentru generarea logaritmilor şi antilogaritmilor într-o bază oarecare.

O altă aplicaţie a logaritmilor obţinuţi prin hardware o constituie aceea din calculatoarele electronice specializate [6], [25], [44], [45]. Astfel, la filtrarea neliniară a unui semnal obţinut prin multiplicarea altor două semnale, ce apare în instalaţiile de radar video [25], [44] este necesară o conversie logaritm-antilogaritm pentru a se putea utiliza apoi o tehnică liniară. Deasemenea, în controlul unor procese industriale [45] este necesară efectuarea unor operaţii aritmetice cu mărimile măsurate sau cu logaritmii acestora. O problemă asemănătoare apare la calculatoare specializate pentru calculul dobînzilor [6]. In toate aceste cazuri problema preciziei nu este esenţială (se admit erori de ordinul procentelor) şi se pot utiliza circuite electronice de generare a logaritmilor şi antilogaritmilor cu complexitate redusă, eventual mai rapide decît dispozitivele de înmulţire şi

INSTITUTUL POLITEHNIC
TIMIŞOARA
BIBLIOTECA CENTRALĂ

BUPT

împărţire [1],[5],[6],[17],[25].

Un loc de utilizare a logaritmilor îl constituie calculatoarele de birou. Marea majoritate a acestora calculează logaritmul natural al unui număr eventual şi logaritmul zecimal, prin metode de aproximare [65], de cele mai multe ori prin desvoltare în serii rapid convergente. Unele calculatoare de birou sînt prevăzute şi cu calculul antilogaritmului natural care reprezintă de fapt calculul funcţiei de forma e^x. Aceste operaţii apelează însă la înmulţiri şi adunări iar în literatura de specialitate nu sînt citate cazuri de utilizare în calculatoarele de birou a unor circuite specializate de generare a logaritmilor şi antilogaritmilor. Calculatoarele de birou pot constitui un loc de aplicare a circuitelor de generare a logaritmilor şi antilogaritmilor dacă acestea sînt suficient de simple şi ieftine întrucît aici viteza de calcul nu constituie principalul obiectiv.

Din cauza faptului că mantisa logaritmului este un număr fracţionar metoda de calcul cu ajutorul logaritmilor nu se poate aplica acolo unde se lucrează cu numere întregi şi se pretinde un rezultat întreg exact. De aceea în calculatoarele numerice universale logaritmii pot fi folosiţi doar în dispozitivul aritmetic cu virgulă mobilă. Aici, dacă baza logaritmului este aceeaşi cu baza sistemului de numeraţie utilizat, caracteristica logaritmului este uşor de determinat din exponentul numărului reprezentat în sistemul cu virgulă mobilă [55], /71/(*

Dacă nu se pretinde un rezultat întreg în calcule, logaritmii se pot utiliza şi în dispozitivele aritmetice cu virgulă fixă prevăzîndu-se însă circuite speciale pentru determinarea caracteristicii [1].

O altă problemă ce apare la utilizarea logaritmilor generaţi prin hardware în calculatoarele numerice o constituie alegerea bazei acestora. Încă de la început [1],[5],[6],[17],[20] s-a observat că logaritmul în baza 2 -logaritmul binar- se poate obţine cel mai simplu cînd numărul ce se logaritmează este reprezentat în sistemul binar. In cazul numerelor disponibile în virgulă fixă [1] mantisa logaritmului este apropiată ca valoare de partea fracţionară a numărului adus prin deplasări în domeniul [1, 2). Cum se vede din fig.1 [1],[3] dacă $1 \leq A < 2$ atunci

$$0 \leq \log_2 A < 1$$

şi

$$\log_2 A \cong A - 1$$

---

*) Bibliografia între bare înclinate constituie lucrări ale autorului tezei.

BUPT

Caracteristica logaritmului unui număr reprezentat în sistemul cu virgulă fixă se determină simplu din cantitatea deplasărilor prin care numărul este adus în domeniul amintit [1].

In cazul unui număr reprezentat în sistemul cu virgulă mobilă, pentru a se putea opera cu logaritmul în baza 2, este necesar ca normalizarea numărului să

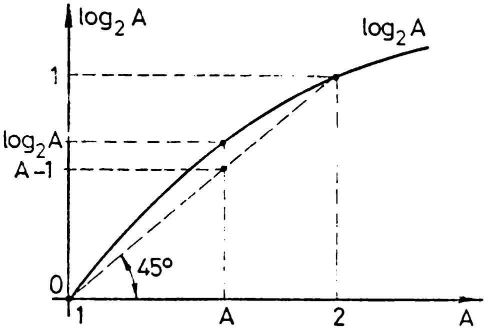


Fig.1. Curba de variaţie a logaritmului binar.

fie făcută în domeniul $[1, 2)$ întrucît numerele normalizate actualmente în calculatoare în domeniul $[\frac{1}{2}, 1)$ au logaritmul binar negativ [1]. In această situaţie caracteristica logaritmului este chiar exponentul numărului. Astfel, dacă numărul reprezentat în sistemul cu virgulă mobilă [55] are forma

$$N = A \times 2^C$$

unde A este mantisa numărului, cuprinsă în domeniul

$$1 \leqslant A < 2$$

iar C este exponentul numărului, atunci :

$$\log_2 N = C + \log_2 A$$

şi cum

$$0 \leqslant \log_2 A < 1$$

rezultă că C reprezintă caracteristica logaritmului binar. Rămîne deci a se rezolva problema generării mantisei logaritmului.

Alte avantaje ale utilizării logaritmilor în baza 2 apar în cazul operaţiilor de ridicare la patrat şi de extragere a rădăcinii patrate cînd sînt necesare doar o logaritmare, o deplasare la stînga sau la dreapta a logaritmului cu o poziţie binară, o antilogaritmare şi o modificare simplă a caracteristicii logaritmului.

In concluzie, pe baza logaritmilor binari generaţi prin hardware cu precizie suficientă şi într-un timp suficient de scurt se poate realiza un dispozitiv aritmetic cu virgulă mobilă care ar putea efectua simplu prin instrucţii "cablate" operaţiile aritmetice de adunare, scădere, înmulţire, împărţire, extragerea rădăcinii patrate, ridicarea la patrat, logaritmarea şi antiloga-

BUPT

ritmarea(în scopul determinării logaritmilor în altă bază, a efectuării ridicării la o putere oarecare şi a extragerii rădăcinii de orice ordin) /71/. Operaţiile din paranteză pot fi efectuate la nevoie chiar şi prin instrucţii "cablate".

Algoritmii pentru calculul logaritmilor şi antilogaritmilor binari dezvoltaţi pînă în prezent care pot asigura precizia necesară într-un calculator numeric universal [19], [24], /58/, [65], /66/ reclamă o cantitate mare de circuite logice /37/. Acest lucru conduce pe de o parte la creşterea costului dispozitivului aritmetic logaritmic iar pe de altă parte la limitarea vitezei acestuia.

In perioada actuală se manifestă însă tot mai mult tendinţa de a se realiza operaţiile aritmetice relativ complicate în calculatoarele numerice cu ajutorul unor "structuri logice celulare" sau "structuri logice iterative"("cellular arrays" sau "iterative arrays"). Prin structură logică celulară se înţelege o schemă logică combinaţională complexă formată din celule identice. O celulă conţine un grup de circuite logice care realizează anumite funcţii logice (ieşirile celulei) pe baza variabilelor de intrare [8], [14], [15], [32]. O structură logică iterativă este o structură logică celulară la care celulele sînt interconectate în mod regulat - uniform [14].

Modul de lucru al structurilor logice celulare care efectuează operaţii aritmetice diferă considerabil de cel al dispozitivelor aritmetice clasice [21]. La intrările structurii logice celulare se aplică nivele de tensiune reprezentînd cifrele binare ale operanzilor iar la ieşirile ei apar nivele de tensiune ce reprezintă cifrele binare ale rezultatului unei operaţii fără altă intervenţie din partea dispozitivului de comandă al calculatorului. Durata operaţiei aritmetice este determinată de timpul de propagare a nivelelor de tensiune prin structura logică celulară. Controlul operaţiilor aritmetice de către dispozitivul de comandă al calculatorului se reduce în acest caz la microoperaţii de transfer al operanzilor la intrările structurii logice celulare şi de preluare a rezultatului de la ieşirile acesteia.

S-au propus pînă în prezent un număr foarte mare de astfel de structuri logice celulare care efectuează operaţiile de înmulţire [10], [13], [14], [16], [18], [21], [22], [31], [38], [39], [41], [42], [48], [51], [54], [63], împărţire [11], [27], [33], [34], [35], [51], [52], [54], [59], [63], extragerea rădăcinii patrate [12], [29], [30], [49], ridicarea la patrat [25], [26], /36/, calculul funcţiilor sinus şi cosinus [53], înmulţirea numerelor complexe [47], compara-

BUPT

rea a două numere [46] şi altele.

Apariţia diferitelor structuri logice celulare şi iterative care efectuează operaţii aritmetice a condus deasemenea la intensificarea cercetărilor în scopul obţinerii logaritmilor şi antilogaritmilor cu ajutorul unor astfel de structuri [19],[24], /37/,/56/,/66/,/70/,/72/.

Avantajele structurilor logice celulare sau iterative sînt următoarele [14],/37/ :

- uşurinţa proiectării logice a unor structuri ce efectuează operaţii complicate,
- extinderea simplă a structurilor logice pentru un număr de cifre binare oricît de mare (flexibilitate),
- uşurinţa interconectării celulelor şi reducerea lungimii interconexiunilor,
- posibilitatea realizării celulelor în formă de circuite integrate de tip MSI,
- posibilitatea realizării unor părţi din structură sau a unor structuri întregi sub formă de circuite integrate de tip LSI sau ELSI (extra LSI),
- densitate mare de circuite deci volum redus a unor scheme,
- posibilitatea testării, diagnosticării şi reparării simple,
- costul relativ redus în cazul folosirii unor circuite integrate standard,
- posibilitatea realizării simple şi rapide a unor operaţii aritmetice complicate,
- necesitatea unui număr restrîns de comenzi din partea dispozitivului de comandă al calculatorului şi deci simplificarea acestuia.

Dintre dezavantajele pe care le au în prezent aceste structuri logice celulare se pot aminti :

- creşterea duratei operaţiilor prin creşterea timpului de propagare la un număr mare de biţi,
- cantitatea mare de celule necesară cînd se operează cu un număr mare de biţi,
- număr mare de terminale ceea ce nu permite întotdeauna utilizarea unor circuite integrate de tip LSI sau ELSI. Este posibil însă ca într-un viitor apropiat o parte din aceste dezavantaje să fie eliminate prin utilizarea unor algoritmi noi sau tehnologii noi [59],[60],[63],[67].

INSTITUTUL POLITEHNIC
TIMIŞOARA
BIBLIOTECA CENTRALĂ

BUPT

Algoritmii cunoscuţi de calcul al logaritmilor şi antilogaritmilor au fost analizaţi de autorul tezei în capitolul I. O primă categorie de algoritmi [1],[5],[6],[25],[62],[44],[17],[45], care se pot implementa cu circuite simple au dezavantajul esenţial al preciziei reduse, de ordinul procentelor şi nu pot fi utilizaţi decît în aplicaţii speciale.

O a doua categorie de algoritmi [7],[19],[24],[65] pot asigura precizia dorită şi pot fi utilizaţi în calculatoarele numerice. Totuşi aceşti algoritmi au încă o serie de dezavantaje importante.

Astfel, al doilea algoritm Dean [7] are următoarele dezavantaje :

- funcţionarea de tip serie (cu registru de deplasare) a dispozitivului propus care nu poate asigura generarea logaritmului sau antilogaritmului într-un timp scurt ,
- proiectarea deosebit de dificilă pentru o cantitate de biţi mare datorită necesităţii de a se minimiza funcţii logice cu acelaşi număr mare de variabile şi datorită necesităţii unui tabel de logaritmi în binar,
- necesitatea reproiectării complete la modificarea cantităţii de biţi ai operandului,
- complexitatea deosebită a circuitelor de comandă ale registrului de deplasare pentru o cantitate mare de biţi.

Al treilea algoritm Dean [19] are următoarele dezavantaje:

- funcţionarea serie-paralel a dispozitivului propus pentru implementarea algoritmului, ceea ce conduce la o durată mare a operaţiei de generare a logaritmilor şi antilogaritmilor, mult mai mare decît durata unei înmulţiri sau împărţiri într-un dispozitiv aritmetic cu virgulă mobilă obişnuit,
- cantitatea mare de circuite necesare - o structură logică celulară de ridicare la patrat şi o structură logică celulară de extragere a ridicării patrate.

Algoritmul Perle [24] are următoarele dezavantaje :

- fiind elaborat pentru o bază oarecare a logaritmului, algoritmul nu este totuşi convergent pentru o bază mai mare decît 2 (fapt sesizat de autorul tezei în /72/),
- durata operaţiilor de logaritmare şi antilogaritmare, după estimările autorului tezei, în cazul implementării algoritmului pe un dispozitiv aritmetic cu virgulă mobilă obişnuit, este de acelaşi ordin de mărime ca durata înmulţirii sau împărţirii şi deci algoritmul nu asigură reducerea timpului de înmulţire

BUPT

sau împărţire în cazul efectuării acestora cu ajutorul logaritmilor,

- necesitatea la calcule a două operaţii diferite - adunare şi scădere,

- efectuează calculele asupra întregului număr, cu parte întreagă şi fracţionară ceea ce măreşte timpul de calcul al logaritmului şi antilogaritmului,

- nu analizează precizia calculului logaritmului sau antilogaritmului ţinînd cont de cantitatea finită de biţi ai numerelor în calculator.

Algoritmul Schmid [65] are următoarele dezavantaje :

- pentru a se aplica şi la generarea antilogaritmului ar fi necesare operaţii de împărţire, care măresc mult durata de generare a antilogaritmului,

- repetarea de mai multe ori a aceluiaşi pas al iteraţiei, ceea ce duce la creşterea duratei operaţiei de logaritmare,

- folosirea logaritmilor negativi a căror valoare este cuprinsă într-un domeniu foarte larg, ceea ce duce la reducerea preciziei de calcul pentru anumite valori ale operanzilor şi la utilizarea ineficientă a circuitelor electronice,

- necesitatea a două tipuri de operaţii : adunare şi scădere la calculul logaritmului.

Rezumînd în ansamblu dezavantajele de mai sus ale algoritmilor şi schemelor de generare a logaritmilor şi antilogaritmilor propuse de alţi autori rezultă următoarele dezavantaje esenţiale comune :

1. Durata mare a operaţiilor de generare a logaritmului şi antilogaritmului, comparabilă sau mult mai mare decît durata operaţiilor de înmulţire sau împărţire dintr-un dispozitiv aritmetic cu virgulă mobilă obişnuit, ceea ce face ca aplicarea logaritmilor la efectuarea înmulţirii sau împărţirii să nu fie avantajoasă din punct de vedere al vitezei de calcul,

2. necesitatea mai multor tipuri de operaţii elementare: transfer între registre, deplasare, adunare, scădere, comparare, generarea unor constante iar la unii algoritmi- ridicarea la patrat, extragerea rădăcinii patrate, împărţire,

3. alegerea nepotrivită a bazei logaritmilor care conduce la necesitatea repetării de un număr de ori a unor paşi din iteraţie,

4. lipsa unei analise a erorilor ce apar în calculul ite-

BUPT

rativ ţinîndu-se cont de cantitatea finită de biţi a numerelor în calculator,

5. implementarea algoritmilor cu circuite secvenţiale, constituite în special din registre, care necesită intervenţii numeroase ale dispozitivului de comandă al calculatorului în procesul de calcul,

6. lipsa unei analize a duratei de generare a logaritmului şi antilogaritmului cu schemele propuse,

7. alegerea nepotrivită a domeniului numerelor ce se logaritmează ceea ce conduce la creşterea duratei de generare a logaritmului şi la complicarea circuitelor electronice.

Autorul tezei şi-a propus deci ca principal scop - elaborarea unui algoritm şi a unor circuite de generare a logaritmului şi antilogaritmului care să elimine în măsura posibilului desavantajele de mai sus şi anume :

- să permită obţinerea unei durate de generare a logaritmului şi antilogaritmului mai mică decît durata înmulţirii şi împărţirii din dispozitivele aritmetice cu virgulă mobilă obişnuită, pentru ca să devină rentabilă înlocuirea acestor operaţii prin operaţii cu logaritmi,

- să necesite mai puţine tipuri de operaţii elementare şi anume numai deplasări, adunări şi comparări,

- să utilizeze logaritmi într-o bază care să nu necesite repetarea unor paşi din iteraţie,

- să permită obţinerea unor erori reduse, a căror limită superioară să fie cunoscută, ţinînd cont de cantitatea finită de biţi a operanzilor,

- să permită implementarea sub formă de structuri logice celulare la care se poate aplica tehnica circuitelor integrate, în care se evită funcţionarea secvenţială şi intervenţia dispozitivului de comandă al calculatorului,

- să permită o extindere simplă, fără reproiectare, la orice cantitate de biţi ai operandului,

- să opereze cu numere normalizate, aşa cum se întîlnesc ele în dispozitivele aritmetice cu virgulă mobilă.

Principalele contribuţii ale autorului tezei la rezolvarea problemei generării logaritmilor şi antilogaritmilor sînt următoarele :

1. Elaborarea unui algoritm nou de calcul iterativ al logaritmului şi antilogaritmului cu următoarele particularităţi /58/:

- operează cu logaritmi în baza 2 şi pune în evidenţă

BUPT

avantajele acestora faţa de logaritmii în altă bază,

- se bazează pe descompunerea în anumiţi factori a numărului ce se logaritmează şi pe descompunerea în anumiţi termeni a numărului ce se antilogaritmează,

- foloseşte numai operaţii de deplasare, adunare şi comparare,

- operează cu numere normalizate în domeniul [1, 2), ceea ce asigură simplitate şi îmbunătăţirea vitezei de calcul,

- nu necesită repetarea aceluiaşi pas din iteraţie ceea ce permite reducerea timpului de calcul şi simplificarea circuitelor electronice,

- permite obţinerea unor erori reduse a căror limită superioară este cunoscută pentru o cantitate dată de biţi ai operanzilor şi a căror provenienţă fiind cunoscută se dau soluţii pentru micşorarea erorilor,

- utilizează constante de forma $1+2^{-i}$ ce au fost calculate pentru $i = 1 \ldots 43$ cu 43 de cifre binare exacte /73/,

- poate fi implementat folosind o structură logică celulară.

Ca punct de plecare la elaborarea acestui algoritm a servit autorului un algoritm cunoscut destinat calculului logaritmilor <u>zecimali</u> cu număr mare de cifre pe baza unor tabele de logaritmi [3].

2. Elaborarea unor circuite de generare a logaritmilor şi antilogaritmilor binari cu următoarele particularităţi /37/ :

- reprezintă structuri logice celulare (deci circuite combinaţionale) ale căror avantaje s-au arătat anterior,

- nu necesită decît aplicarea biţilor operandului la intrări sub formă de nivele logice iar după un anumit timp cunoscut se obţin la ieşiri biţii rezultatului,

- se pot extinde foarte simplu la orice lungime de operand,

- realizează operaţia de generare a logaritmului şi antilogaritmului într-un timp redus, ceea ce permite obţinerea unui avantaj în viteză la înlocuirea înmulţirii şi împărţirii cu operaţii prin logaritmi într-un dispozitiv aritmetic cu virgulă mobilă,

- structurile logice celulare pot funcţiona în regim asincron, exploatîndu-se faptul că durata calculului depinde de valoarea numărului ce se logaritmează sau antilogaritmează, cînd durata medie de generare a logaritmului şi antilogaritmului scade de cca. 2 ori,

- pot fi realizate prin integrare pe scară medie la nivel

BUPT

de grupe de celule.

3. Elaborarea unui program în FORTRAN pentru simularea pe calculator numeric a structurilor logice celulare /64/ bazat pe principiul urmăririi selective care furnizează date despre durata operației efectuate de structuri, permite verificarea funcționării corecte a schemei logice și sesizarea modului în care este influențată durata operațiilor de către schema logică a celulelor.

4. Elaborarea unor algoritmi noi de funcționare a unui dispozitiv aritmetic cu virgulă mobilă logaritmic /71/ și concepția unui astfel de dispozitiv original, avînd următoarele particularități :

- poate realiza 8 operații aritmetice: adunare, scădere, înmulțire, împărțire, extragere a rădăcinii patrate, ridicare la patrat, logaritmare și antilogaritmare în baza 2,

- durata unei înmulțiri sau împărțiri de 2,5 respectiv de 3 ori mai mică decît la un dispozitiv aritmetic obișnuit realizat cu același tip de circuite integrate,

- poate funcționa în regim asincron pentru cele 6 operații ce folosesc logaritmi mărindu-se astfel în medie cu 30% viteza de calcul în cadrul acestor operații,

- durata operațiilor de ridicare la patrat și extragere a rădăcinii patrate este mai mică decît a operațiilor de înmulțire și împărțire fără să se aducă o complicare a circuitelor de comandă a dispozitivului aritmetic,

- dispozitivul aritmetic poate fi realizat ca opțiune la un calculator numeric universal necesitînd de la acesta doar un număr redus de comenzi referitoare la instrucțiile ce trebuie efectuate și la transferul operanzilor în și din memorie.

Inafara acestora, autorul a mai adus următoarele contribuții în domeniul algoritmilor și circuitelor de generare a logaritmilor și antilogaritmilor :

- elaborarea unui algoritm nou de generare a logaritmilor și antilogaritmilor binari bazat pe înmulțirea și împărțirea cu constante cunoscute /66/,

- elaborarea unei scheme secvențiale, ce utilizează o structură logică celulară de înmulțire și împărțire, care implementează algoritmul citat mai sus /66/ și este utilizabilă într-un calculator de birou sau în calculatoare specializate,

- elaborarea unei structuri logice celulare rapide de antilogaritmare în baza 2, bazată pe înmulțirea unor constante în funcție de valorile biților logaritmului /56/.

BUPT

- elaborarea unei metode de proiectare pentru generatoare de constante de forma $2^{2^{-i}}$ şi $1+2^{-i}$ utilizate în cadrul algoritmilor dezvoltaţi de autor cînd aceştia se aplică în dispozitive secvenţiale de generare a logaritmilor şi antilogaritmilor binari /7o/,
- corectarea şi completarea algoritmului Perle /72/,
- calculul duratei operaţiei de generare a logaritmului şi antilogaritmului prin algoritmii cunoscuţi [19], [24],
- elaborarea unei noi structuri logice celulare de ridicare la patrat /36/ destinată algoritmului Dean [19].

In total, referitor la problemele cercetate în teză autorul are publicate sau în curs de publicare 8 lucrări şi comunicate 2 lucrări.

Algoritmii noi de logaritmare şi antilogaritmare propuşi de autorul tezei se pot deasemenea utiliza la întocmirea unor subprograme pentru calculatoare numerice /58/,/66/ dar această problemă nu a fost studiată în cadrul tezei.

Teza de doctorat conţine cinci capitole.

In Capitolul I se analizează critic algoritmii şi circuitele logice de generare a logaritmilor şi antilogaritmilor propuse pînă în prezent de diferiţi autori. Acestea sînt grupate în două categorii în funcţie de precizia pe care o pot asigura şi deci în funcţie de destinaţia lor. Autorul tezei aduce o serie de completări la algoritmi privind calculul duratei de generare a logaritmului şi antilogaritmului sau referitoare la convergenţa unor iteraţii. Autorul descoperă o greşeală în algoritmul Perle [19] pe care a sesizat-o şi în literatura de specialitate /72/ şi propune o soluţie pentru rezolvarea corectă a problemei.

In Capitolul II se prezintă contribuţia autorului tezei la elaborarea de noi algoritmi aplicabili la generarea prin hardware a logaritmilor şi antilogaritmilor binari. Autorul subliniază aici avantajele alegerii bazei 2 pentru logaritmi.

Se prezintă mai întîi algoritmul bazat pe înmulţirea şi împărţirea cu constante apoi algoritmul bazat pe descompunerea în factori a numărului ce se logaritmează. Sînt analizate amănunţit erorile absolute ce pot să apară în calculul iterativ şi posibilităţile de reducere, ţinîndu-se cont de cantitatea finită de biţi a numerelor din calculator. O serie de constante de forma $2^{2^{-i}}$ şi $1+2^{-i}$ ce apar în cadrul algoritmilor au fost calculate aici cu ajutorul calculatorului electronic cu 43 cifre binare exacte.

BUPT

In Capitolul III, cel mai extins din teză, sînt prezentate circuitele electronice propuse pentru implementarea algoritmilor noi elaboraţi de autor şi anume :

- un dispozitiv secvenţial incluzînd o structură logică celulară de înmulţire şi împărţire, care generează logaritmul şi antilogaritmul binar prin înmulţiri şi împărţiri cu constante,
- un generator de constante de forma $2^{2^{-i}}$ pentru dispozitivul citat mai sus,
- o structură logică celulară de generare a antilogaritmilor binari bazată pe înmulţiri de constante,
- structuri logice celulare de generare a logaritmului binar bazate pe descompunerea în factori de forma $1+2^{-i}$,
- structuri logice celulare de generare a antilogaritmului binar bazate pe descompunerea în termeni de forma $\log_2(1+2^{-i})$,
- structuri celulare comasate de generare a logaritmului şi antilogaritmului binar,
- structuri logice celulare separate şi comasate cu anticipare pentru creşterea vitezei de generare a logaritmului şi antilogaritmului binar,
- structuri logice celulare comasate cu anticipare asincrone de generare a logaritmului şi antilogaritmului binar.

In cadrul fiecăreia din problemele tratate aici se studiază schemele celulelor, durata operaţiilor şi posibilităţile de creştere a vitezei de calcul a structurilor. Se determină de fiecare dată durata maximă şi minimă a operaţiei. Prin utilizarea unor observaţii se concepe o structură rapidă asincronă care exploatează faptul că durata operaţiilor depinde de valoarea operandului. Timpul de generare a logaritmului şi antilogaritmului este mult mai mic decît durata operaţiilor de înmulţire şi împărţire dintr-un dispozitiv aritmetic cu virgulă flotantă obişnuit realizat cu acelaşi tip de circuite integrate [74].

In Capitolul IV se prezintă programele de simulare a structurilor logice celulare şi rezultatele obţinute la simulare. Simularea a fost necesară deoarece structurile logice de generare a logaritmilor şi antilogaritmilor sînt complexe, astfel încît nici experimentarea lor la scară redusă şi nici studiul analitic pentru o cantitate redusă de biţi ai numerelor nu ar fi condus la concluzii extrapolabile pentru o cantitate mare de biţi. In urma simulării s-au putut preciza cazurile cele mai defavorabile, cînd durata operaţiilor de generare a logaritmului şi antilogaritmului este maximă. Deasemenea, în urma simulării s-au stabilit schemele

BUPT

logice ale celulelor, date în Capitolul III care asigură cea mai mare viteză de calcul.

In Capitolul V se prezintă algoritmii noi de funcţionare a unui dispozitiv aritmetic cu virgulă mobilă logaritmic care efectuează 8 operaţii aritmetice : adunare, scădere, înmulţire, împărţire, ridicare la patrat, extragerea rădăcinii patrate, logaritmare şi antilogaritmare în baza 2. Pe baza algoritmilor se concepe apoi un dispozitiv aritmetic cu virgulă mobilă logaritmic original, care efectuează înmulţirea de 2,5 ori mai repede, iar împărţirea de 3 ori mai repede decît un dispozitiv aritmetic cu virgulă mobilă obişnuit realizat cu acelaşi tip de circuite [74], [76]. Dispozitivul propus în teză prezintă deasemenea avantajul că poate folosi aceleaşi registre ca şi dispozitivul cu virgulă fixă al unui calculator întrucît nu necesită elemente separate pentru prelucrarea exponenţilor.

In încheiere se prezintă concluzii în legătură cu rezultatele cercetărilor şi posibilităţile de aplicare. Sînt enumerate apoi contribuţiile autorului tezei în domeniul aritmeticii şi al structurii dispozitivelor aritmetice ale calculatoarelor numerice.

Pentru îndrumarea şi sprijinul permanent acordat pe parcursul activităţii în cadrul doctoratului şi la elaborarea tezei aduc pe această cale călduroase mulţumiri conducătorului ştiinţific - tovarăşului prof.dr.ing.Alexandru Rogojan, şeful Catedrei de calculatoare electronice.

BUPT

CAPITOLUL I

ALGORITMI SI CIRCUITE CUNOSCUTE DE GENERARE A LOGARITMILOR SI ANTILOGARITMILOR BINARI

Algoritmii cunoscuţi pînă în present de generare prin hardware a logaritmilor şi antilogaritmilor binari pot fi împărţiţi în două categorii, în funcţie de precizia pe care o pot asigura, deci în funcţie de destinaţie.

Din prima categorie fac parte algoritmii ce generează logaritmii şi antilogaritmii cu o eroare de ordinul procentelor şi care au fost denumiţi mai jos :

- algoritmul Mitchell [1],
- algoritmul Combet [5],
- primul algoritm Dean [6],
- algoritmul Hall [25],
- algoritmul Marino [62]
- algoritmul Jullien [44],
- algoritmul Andrews [17],
- algoritmul Nicand [45],

Din cea de a doua categorie fac parte algoritmii ce pot genera logaritmii şi antilogaritmii binari cu precizia dorită prin simpla extindere a circuitelor logice la un număr corespunzător de biţi. Aceşti algoritmi au fost denumiţi în continuare :

- al doilea algoritm Dean [7],
- al treilea algoritm Dean [19],
- algoritmul Perle [24],
- algoritmul Schmid [65].

Cei doi algoritmi propuşi şi dezvoltaţi de autorul tezei fac parte deasemenea din această a doua categorie însă ei vor fi prezentaţi în capitolul următor.

1.1. Algoritmi de precizie redusă

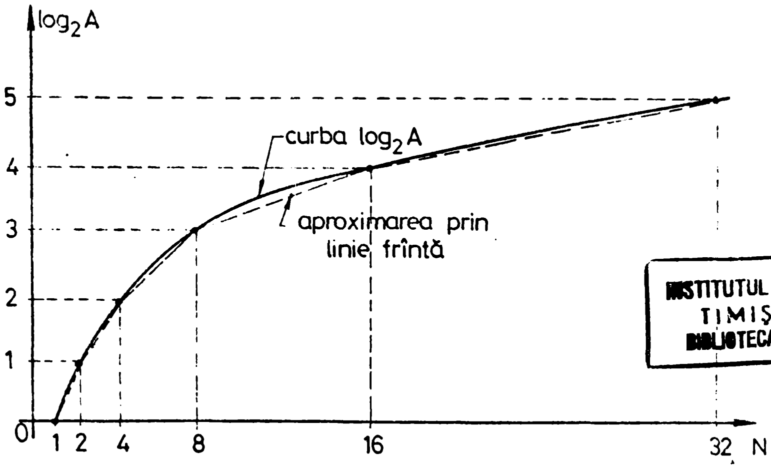
1.1.1. Algoritmul Mitchell

J.N.Mitchell [1] pune pentru prima dată problema utilizării logaritmilor binari la efectuarea operaţiilor de înmulţire şi împărţire. Algoritmul său se bazează pe aproximarea curbei logaritmului binar cu o linie frîntă ce trece prin punctele de pe curbă care au logaritmul binar un număr întreg (fig.1-1).

Avînd valorile aproximative ale logaritmilor binari pentru numerele 1-16 daţi de linia frîntă (Tabelul 1-1) Mitchell a dedus

BUPT

următoarele :



INSTITUTUL POLITEHNIC
TIMIŞOARA
BIBLIOTECA CENTRALĂ

Fig.1-1 Curba logaritmului binar

Tabelul 1-1

N în zecimal	N în binar	$\log_2 N$ în binar
1	00001	000,0000
2	00010	001,0000
3	00011	001,1000
4	00100	010,0000
5	00101	010,0100
6	00110	010,1000
7	00111	010,1100
8	01000	011,0000
9	01001	011,0010
10	01010	011,0100
11	01011	011,0110
12	01100	011,1000
13	01101	011,1010
14	01110	011,1100
15	01111	011,1110
16	10000	100,0000

1. Caracteristica logaritmului este determinată de poziţia primului bit egal cu 1 din stînga numărului (scris apăsat în Tabelul 1-1). Cu alte cuvinte caracteristica este egală cu puterea lui 2 corespunzătoare acestei poziţii binare.

2. Valoarea aproximativă a mantisei logaritmului reprezintă partea numărului aflată în dreapta bitului 1 cel mai din stînga (scris apăsat în tabel).

In acest fel obţinerea logaritmilor binari aproximativi şi aplicarea lor într-un dispozitiv aritmetic cu virgulă fixă este simplă (fig.1-2).

In registrele de deplasare la stînga A şi B se găsesc cele două numere ce trebuie înmulţite sau împărţite. Numerele au 8 biţi : se obţine astfel o caracteristică cel mult egală cu 7 pentru a cărei reprezentare sînt necesare 3 poziţii binare : X_1, X_2, X_3 şi Y_1, Y_2, Y_3 în registrele C şi D ale logaritmilor. Aceste două grupe de celule sînt

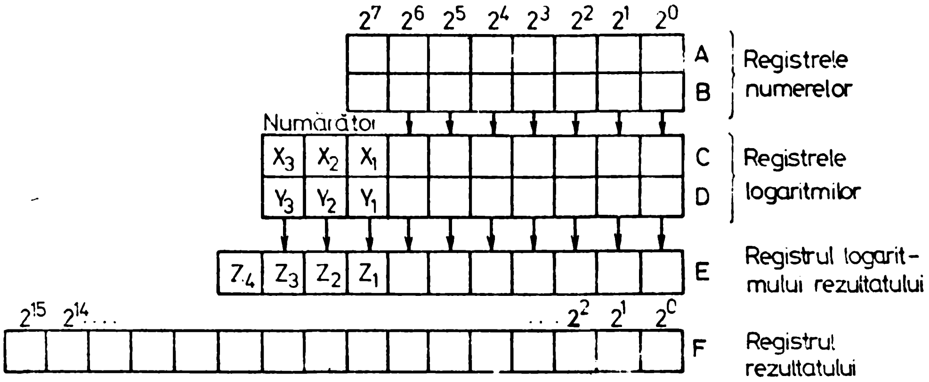


Fig.1-2. Dispozitivul aritmetic logaritmic Mitchell.

conectate ca numărătoare ce conţin fiecare în starea iniţială numărul 111.

Conţinutul registrelor A şi B se deplasează la stînga pînă cînd în poziţia 2^7 apare cifra 1. La fiecare deplasare din conţinutul numărătorului corespunzător se scade un 1. La sfîrşitul deplasărilor conţinuturile numărătoarelor vor reprezenta caracteristicile logaritmilor numerelor din registrele A şi B.

Conţinuturile rangurilor 2^0 pînă la 2^6 ale registrelor A şi B se transferă apoi simultan în registrele C şi D unde se găsesc acum valorile aproximative ale logaritmilor numerelor.

Se efectuează în continuare operaţia de adunare sau scădere a logaritmilor şi logaritmul rezultat se trimite în E. Acesta poate avea caracteristica cel mult egală cu 15 (7+7 plus un eventual transport la adunarea mantiselor.

Pentru efectuarea operaţiei de antilogaritmare se înscrie un 1 în poziţia registrului F indicată de conţinutul părţii $Z_1 \ldots Z_4$ a registrului E. Deasemenea, conţinutul părţii rămase din registrul E se înscrie în registrul F imediat în dreapta acestui 1. Registrul F va conţine după aceste operaţii rezultatul.

Obţinerea logaritmilor şi antilogaritmilor binari după acest algoritm este simplă însă eroarea cu care se deduc aceştia şi deci eroarea rezultatului înmulţirii sau împărţirii este mare.

Pentru analiza erorii algoritmului numărul binar a fost scris [1] în forma :

$$N = 2^k(1+x)$$

unde k reprezintă caracteristica logaritmului binar iar x, care este cuprins în domeniul :

BUPT

$$0 \leqslant x < 1 \qquad (1-2)$$

reprezintă valoarea aproximativă a mantisei logaritmului binar. Se poate observa că numărul fracţionar x se obţine eliminînd prima cifră 1 din stînga a numărului N şi punînd virgulă după această poziţie.

Eroarea absolută la logaritmarea unui număr este :

$$\Delta L = \log_2(1+x) - x \qquad (1-3)$$

Ea are valoarea maximă $\Delta L_{max} = 0{,}08639$ pentru $x = 0{,}44269$ (eroarea relativă cca. 16%) şi este nulă la capetele domeniului lui x [1].

In urma operaţiilor de înmulţire sau împărţire, în anumite situaţii, vor rezulta erori relative maxime de -11,1% respectiv +12,5%.

Autorul lucrării [1] a găsit o posibilitate de reducere a erorii relative în cadrul operaţiei de înmulţire prin generarea şi adăugarea unei corecţii. Se realizează astfel o eroare relativă maximă a produsului de -2,8%. Dacă se repetă corecţia de un număr mare de ori se poate face eroarea oricît de mică. Aceasta conduce însă la creşterea exagerată a timpului de execuţie a operaţiei. In cazul împărţirii autorul lucrării [1] nu a găsit o posibilitate de corecţie simplă.

Lucrarea [1] are meritele :

- de a fi semnalat anumite particularităţi ale logaritmilor binari care simplifică circuitele logice de generare a acestora,
- de a fi semnalat că domeniul cel mai potrivit pentru numerele binare ce se logaritmează este

$$1 \leqslant N < 2 \qquad (1-4)$$

unde $N = 1 + x$ iar $0 \leqslant x < 1$

- de a fi prezentat modul de obţinere a caracteristicii logaritmilor binari în cazul numerelor reprezentate în sistemul cu virgulă fixă.

Eroarea ce apare în operaţia de logaritmare şi antilogaritmare fac însă ca algoritmul Mitchell să fie inacceptabil pentru calculatoare numerice universale. Timpul lung necesar pentru determinarea caracteristicii logaritmului în cazul numerelor reprezentate în virgulă fixă nu poate fi evitat.

INSTITUTUL POLITEHNIC TIMIŞOARA BIBLIOTECA CENTRALĂ

1.1.2. Algoritmul Combet.

H.Combet, Van Zonneveld şi L.Verbeek [5] prezintă un algoritm de generare a logaritmilor binari cu precizie mai bună decît algoritmul Mitchell, care constă într-o aproximare mai bună

BUPT

prin segmente de linie dreaptă a curbei logaritmice (Fig.1-3). Pentru realizarea simplă a dispozitivului de logaritmare domeniul pentru numărul fracţionar x este împărţit în patru intervale egale. Valoarea aproximativă a logaritmului numărului 1+x se obţine prin operaţiile simple prezentate în Tabelul 1-2, unde $\bar{x}$ reprezintă complementul faţă de 1 al lui x.

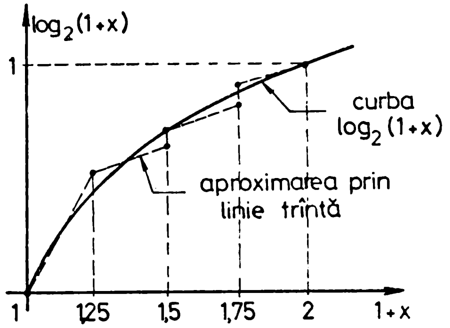


Fig.1-3. Aproximarea curbei logaritmului în algoritmul Combet.

Tabelul 1-2

Intervalul	Logaritmul aproximativ
$0 \leq x < \frac{1}{4}$	$x + \frac{5}{16}x$
$\frac{1}{4} \leq x < \frac{1}{2}$	$x + \frac{5}{64}$
$\frac{1}{2} \leq x < \frac{3}{4}$	$x + \frac{1}{8}\bar{x} + \frac{3}{128}$
$\frac{3}{4} \leq x < 1$	$x + \frac{1}{4}\bar{x}$

Determinarea logaritmului aproximativ după această metodă implică : decizii logice pentru stabilirea intervalului, deci a tipului corecţiei, deplasări şi adunări. Schema bloc a dispozitivului de generare a logaritmilor binari după acest algoritm este prezentată în fig.1-4.

Schema conţine un circuit de bază pentru determinarea logaritmului aproximativ după metoda Mitchell, compus dintr-un registru de deplasare R şi un numărător C. Registrul R conţine iniţial numărul N pe care îl deplasează spre stînga pînă ce bitul 1 cel mai semnificativ părăseşte registrul. Numărul rămas în registrul R va fi

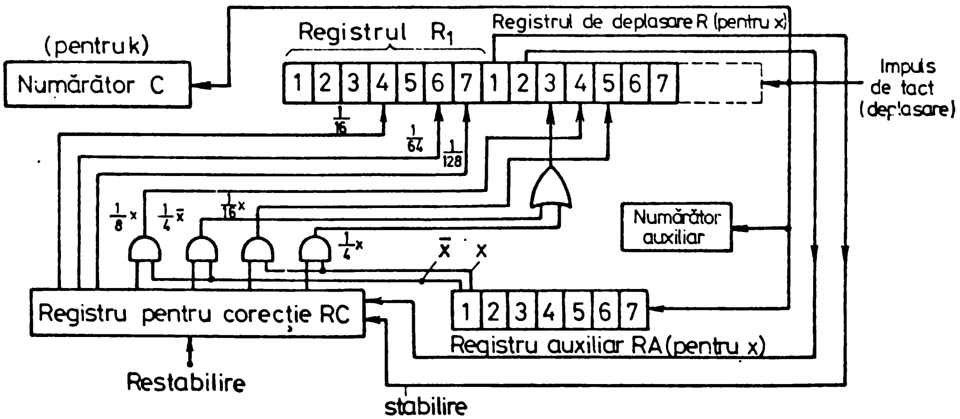


Fig.1-4. Dispozitivul de generare a logaritmilor după algoritmul Combet.

BUPT

tocmai x. Numărătorul C care conține inițial un număr egal cu cantitatea biților registrului R, își reduce conținutul cu 1 la fiecare deplasare și va conține în final caracteristica k a lui $\log_2 N$.

Pentru adăugarea corecțiilor date în tabelul 1-2 numărul x este trimis și în registrul auxiliar RA. Corecțiile care sînt funcție de x au fost puse în forma $\frac{1}{2^p}x$, $\frac{1}{2^p}\bar{x}$ sau $\frac{1}{2^p}x + \frac{1}{2^p}x$ iar adunarea lor la numărul x aflat în registrul R constă în a aduna pe x sau $\bar{x}$ deplasat corespunzător. Adunarea corecțiilor se face însă în serie, prin deplasarea pas cu pas spre stînga a numerelor din registrele R și RA. La un moment dat este adunat numai bitul din poziția cea mai semnificativă în poziția corespunzătoare a registrului R, ceea ce include deplasarea cu p sau q poziții a acestui bit. După operația de corecție mantisa se găsește în registrul R_1. Aici se efectuează corecția cu termenul constant (Tabelul 1-2) care deasemenea este pus în forma $\frac{1}{2^r} + \frac{1}{2^s}$. Se adună deci la mantisă 1-uri în pozițiile r și s ale registrului R_1.

Registrul de corecție RC furnizează corecția necesară în funcție de valoarea primilor doi biți ai mantisei necorectate x din registrul R (în funcție de acești biți se poate stabili intervalul necesar din cele patru posibile).

Dispozitivul din fig.1-4 a fost realizat experimental [2] pentru numere cu 22 biți, corecția făcîndu-se însă numai asupra primilor 7 biți. Timpul maxim de obținere a logaritmului a fost de 60 perioade de tact. In acest dispozitiv - destinat unui periodmetru numeric - nu a fost prevăzută antilogaritmarea dar ea ar fi posibilă prin utilizarea unei tehnici similare.

Eroarea absolută maximă ce apare la logaritmare după această metodă este $\Delta L = 0{,}013$ pentru $x = 0{,}625$ și $x = 0{,}687$ (eroare relativă cca 2%). Ea se poate reduce însă de două ori prin dublarea numărului de intervale ale domeniului lui x ceea ce complică circuitele logice pentru generarea logaritmului.

Algoritmul Combet prezintă deci ca dezavantaj principal acela că eroarea logaritmului binar generat este mare, nu se poate reduce prin creșterea cantității de biți a numerelor și nu este acceptabilă pentru calculatoare numerice universale. Deasemenea, din cauza deplasărilor pentru corecție, durata în care se generează logaritmul este mare.

1.1.3. Primul algoritm Dean

In [6] se prezintă o metodă similară cu metoda Combet, aproximînd curba logaritmului prin două segmente de dreaptă

BUPT

(fig.1-5). Aici s-a considerat numărul x cuprins în intervalul

$$1 \leqslant x < 2 \qquad (1-4)$$

In tabelul 1-3 se arată ecuațiile dreptelor în cele două intervale.

Tabelul 1-3

Intervalul	Ecuația
A $1 < x < 1{,}5$	$(x-1)+\frac{x-1}{4}$
B $1{,}5 < x < 2$	$(x-1)-\frac{x}{4}$

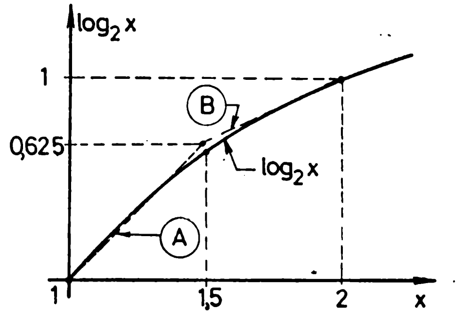


Fig.1-5. Aproximarea curbei logaritmului în algoritmul Dean.

Pentru generarea logaritmului binar aproximativ cu această metodă în intervalul A se utilizează schema serie din fig.1-6.

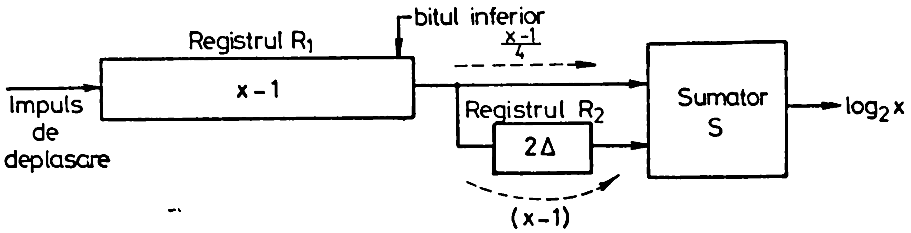


Fig.1-6. Schema de generare a logaritmului după algoritmul Dean (intervalul A).

Schema conține un registru de deplasare R_1, un registru de deplasare cu 2 biți R_2 și un sumator serie S. Pentru generarea logaritmului binar când numărul x se află în intervalul B se utilizează schema serie din fig.1-7, asemănătoare cu prima și cu care se poate deci comasa.

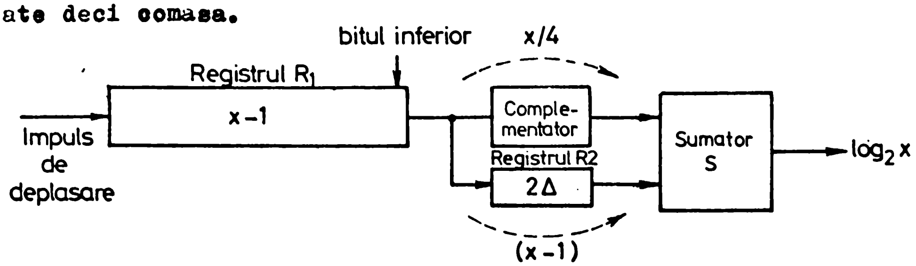


Fig.1-7. Schema de generare a logaritmului după algoritmul Dean (intervalul B).

Pentru generarea antilogaritmului autorul sugerează o tehnică similară [6].

Eroarea absolută maximă ce apare la calculul logaritmilor

BUPT

este $\Delta L_{max} = -0,0469$ şi apare în punctul $x = 1,5$ (eroare relativă de 8,11 %). Pentru reducerea erorii acestui algoritm acelaşi autor mai propune [6] o împărţire a domeniului lui x în trei intervale. Eroarea relativă a logaritmului astfel generat rămîne însă de ordinul procentelor.

Circuitele necesare la generarea logaritmului binar sînt simple. Se poate alcătui uşor o schemă cu funcţionare paralelă suficient de rapidă.

Dezavantajul principal al acestui algoritm îl constituie eroarea mare a logaritmului, ceea ce îl face utilizabil numai în aplicaţii speciale.

1.1.4. Algoritmul Hall.

In [25] E.L.Hall, D.D.Lynch şi S.J.Dwyer au prezentat un algoritm de generare a logaritmilor şi antilogaritmilor binari asemănător cu algoritmul Combet. Si aici domeniul numărului fracţionar x este împărţit în aceleaşi patru intervale însă aproximarea liniară în fiecare interval se face cu condiţia obţinerii unei erori medii practice minime. In acest fel se obţine şi o eroare absolută mai mică.

In Tabelul 1-4 se prezintă intervalele numărului fracţionar x şi ecuaţiile segmentelor de dreaptă ce aproximează curba logaritmului în cadrul operaţiilor de generare a logaritmilor şi antilogaritmilor. S-a notat aici $\bar{x} = 1 - x$.

Tabelul 1-4

Interval	Aproximarea: Calculul logaritmului	Aproximarea: Calculul antilogaritmului
$0 \leq x < 1/4$	$x + \frac{37}{128} x + \frac{1}{128}$	$x + \frac{1}{4} \bar{x} + \frac{3}{4}$
$1/4 \leq x < 1/2$	$x + \frac{3}{64} x + \frac{1}{16}$	$x + \frac{13}{128} \bar{x} + \frac{55}{64}$
$1/2 \leq x < 3/4$	$x + \frac{7}{64} \bar{x} + \frac{1}{32}$	$x + \frac{9}{128} x + \frac{7}{8}$
$3/4 \leq x < 1$	$x + \frac{29}{128} \bar{x}$	$x + \frac{35}{128} x + \frac{23}{32}$

Prin acest algoritm se realizează la generarea logaritmului o eroare medie patratică cel mult egală cu $\Delta L^2_{med} = 3,33.10^{-6}$ şi o eroare absolută maximă $\Delta L = +0,00994$ (eroare relativă cca 2%). Erorile la generarea antilogaritmului sînt mai mici. Eroarea relativă maximă la efectuarea unei înmulţiri este de cca 2%. Din acest motiv algoritmul nu se poate utiliza în calculele de precizie şi

INSTITUTUL POLITEHNIC TIMIŞOARA ... CENTRALĂ
BUPT

numai în aplicaţii speciale ca cele enumerate în introducere [25].

1.1.5. Algoritmul Marino

Analizînd curba erorii absolute R din algoritmul Mitchell, dată în fig.1-8, Marino [62] găseşte relaţii simple pentru efectuarea unei corecţii care să îmbunătăţească precizia.

Logaritmul se calculează deci cu relaţia :

$$\log_2(1+x) = x + R(x) \qquad (1\text{-}5)$$

unde eroarea absolută R(x) se aproximează parabolic în mod diferit în intervalele 0-0,5 şi 0,5-1.

Prima parabolă interpolează punctele x=0; 0,25; 1 şi are ecuaţia :

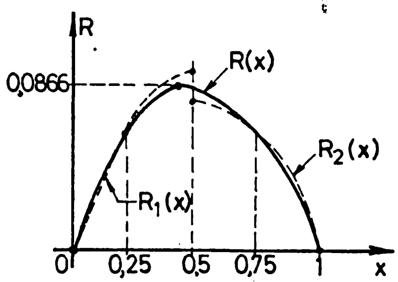


Fig.1-8 Curba erorii absolute aproximată de Marino.

$$R_1(x) = 4(2^{-4}+2^{-5})\ (x-x^2)$$

pentru $0 \leqslant x < 0,5$ (1-6)

iar a doua parabolă interpolează punctele cu x = 0; 0,75; 1, avînd ecuaţia :

$$R_2(x) = 4(2^{-4}+2^{-6})\ (x-x^2) \quad \text{pentru} \quad 0,5 \leqslant x < 1 \qquad (1\text{-}7)$$

Termenul x^2 este determinat tot pe cale de aproximaţie cu relaţiile :

$$x^2 \cong 2^{-2j_1}(1+2\,x_1)+2^{-2(j_1+j_2)}\,(1+4\,x_2),\ \text{pentru}\ x<0,5, \qquad (1\text{-}8)$$

$$x^2 \cong 2^{-2j_1}(1+2\,x_1)+2^{-2(j_1+j_2)}\,(1+2\,x_2),\ \text{pentru}\ x \geqslant 0,5, \qquad (1\text{-}9)$$

în care j_1 este ordinul celui mai semnificativ bit 1 din partea fracţionară a lui x iar x_1 este partea fracţionară subunitară ce se obţine dacă se plasează virgula în dreapta acestui 1 al lui x. In mod asemănător se definesc j_2 şi x_2 pentru numărul x_1.

In cadrul algoritmului sînt necesare în total patru operaţii de adunare (înmulţirea cu factorul binar din faţa parantezei se face prin deplasare şi adunare). Eroarea absolută maximă a lui $\log_2(1+x)$ este în acest caz : $\Delta L_{max} = 0,004$ pentru x=0,125 (eroarea relativă cca 3,2%), ceea ce constituie principalul dezavantaj al algoritmului. Schema necesară la generarea logaritmului cu acest algoritm este relativ simplă [62]. Aceeaşi corecţie se poate aplica şi la generarea antilogaritmului dar cu semn schimbat.

1.1.6. Algoritmul Jullien.

G.A.Jullien [44] prezintă un algoritm de calcul al loga-

BUPT

ritmilor binari, utilizabil atît pentru calcule pe calculator numeric cît şi pentru generarea logaritmilor prin hardware. Algoritmul este destinat numerelor întregi şi se exprimă prin relaţia:

$$\log_2 N \cong n + \left(\frac{N}{2^n} - 1\right) \qquad (1-10)$$

unde n este definit prin condiţia :

$$2^n < N < 2^{n+1} \qquad (1-11)$$

Eroarea la calculul logaritmului este mare pentru $N < 10$ şi este $<1\%$ pentru $N > 100$. Din această cauză algoritmul propus de Jullien este utilizat doar în aplicaţii speciale [44].

1.1.7. Algoritmul Andrees

In [17] C.A.Andrews propune un algoritm pentru calculul logaritmului binar care asigură 4 biţi exacţi în mantisă. Caracteristica logaritmului şi numărul fracţionar x se determină la fel ca în algoritmul Mitchell [1]. In Tabelul 1-5 se prezintă cei 4 biţi exacţi ai mantisei logaritmului pentru toate valorile pe care le poate lua un număr cu 4 biţi.

Tabelul 1-5

x în binar	$\log_2(1+x)$ în binar
0,0000	0,0000
0,0001	0,0001
0,0010	0,0010
0,0011	0,0011
0,0100	0,0101
0,0101	0,0110
0,0110	0,0111
0,0111	0,1000
0,1000	0,1001
0,1001	0,1010
0,1010	0,1011
0,1011	0,1100
0,1100	0,1100
0,1101	0,1101
0,1110	0,1110
0,1111	0,1111

Analiza tabelului arată că dacă numărul x conţine numai cifre 0 sau 1 în poziţiile 2^{-1} şi 2^{-2} atunci :

$$\log_2(1+x) = x \qquad (1-12)$$

iar dacă cifrele din poziţiile 2^{-1} şi 2^{-2} ale lui x diferă între ele atunci :

$$\log_2(1+x) = x+2^{-4} \qquad (1-13)$$

Pentru realizarea acestui algoritm sînt necesare, înafară de circuitele pentru stabilirea caracteristicii, circuite logice simple [17] care să ia o decizie logică şi să adauge eventual un 1 în poziţia 2^{-4} a numărului fracţionar x.

Dezavantajul principal al acestui algoritm este acela că el nu se poate extinde la un număr mare de cifre binare rămînînd o metodă de precizie redusă, utilizabilă într-un număr restrîns de aplicaţii speciale.

BUPT

1.1.8. Algoritmul Nicaud

In lucrarea [45] J.D.Nicaud şi R.Dessoulavy au descris o metodă pentru generarea logaritmului în orice bază pentru un domeniu larg de numere, cu o precizie limitată numai de timpul de calcul şi de lungimea registrelor.

Pentru calculele logaritmului unui număr dat x se determină o valoare u^n apropiată de x. Astfel n reprezintă aproximarea lui $\log_u x$. Se pot realiza circuite care să genereze treptat puterile lui u şi să compare pe rînd fiecare valoare u^n cu numărul x dat. Cînd numărul x este atins valoarea n reprezintă cu aproximaţie pe $\log_u x$.

Calculul antilogaritmului este simplu deoarece antilogaritmul unui număr m este egal cu u^m. Circuitele trebuie să genereze deasemenea puterile u^n şi să compare fiecare valoare n cu numărul m. Cînd numărul m este atins, cantitatea u^n reprezintă antilogaritmul aproximativ.

Ridicarea la putere a bazei u a logaritmului se efectuează prin înmulţiri dar alegînd o anumită valoare particulară pentru u se poate reduce înmulţirea la o deplasare şi o adunare [45]. Astfel, dacă se alege ca bază valoarea $u = 1+2^{-k}$, unde k este un întreg pozitiv, produsul dintre un rezultat anterior r şi u devine :

$$ru = r(1+2^{-k}) = r+r.2^{-k} \qquad (1-14)$$

deci produsul se transformă într-o deplasare la dreapta cu k poziţii şi o adunare. Adunarea se poate efectua în paralel sau în serie.

In fig.1-9 se prezintă un sistem serie de generare a logaritmului care utilizează un registru de deplasare la dreapta R_1, un sumator serie, un registru R_2 pentru numărul ce se logaritmează, un comparator serie, un numărător C şi circuite de comandă.

Deplasarea la dreapta cu k poziţii se face folosind o ieşire din poziţia anterioară ultimilor k biţi ai registrului R_1. In fiecare ciclu de i+j impulsuri de deplasare (egal cu cantitatea de biţi a numărului $(1+2^{-k})^n$)se multiplică conţinutul lui R_2 cu $1+2^{-k}$, se adună la conţinutul numărătorului C un 1 şi se compară conţinuturile registrelor R_1 şi R_2.

Cînd conţinutul registrului R_1 devine egal sau mai mare decît cel al registrului R_2 în numărătorul C se va găsi logaritmul în baza $(1+2^{-k})$ al numărului din registrul R_2.

Pentru calculul antilogaritmului numărului introdus în registrul R_2,la intrarea comparatorului trebuie adus în serie conţinutul registrului R_2 şi al numărătorului C.

BUPT

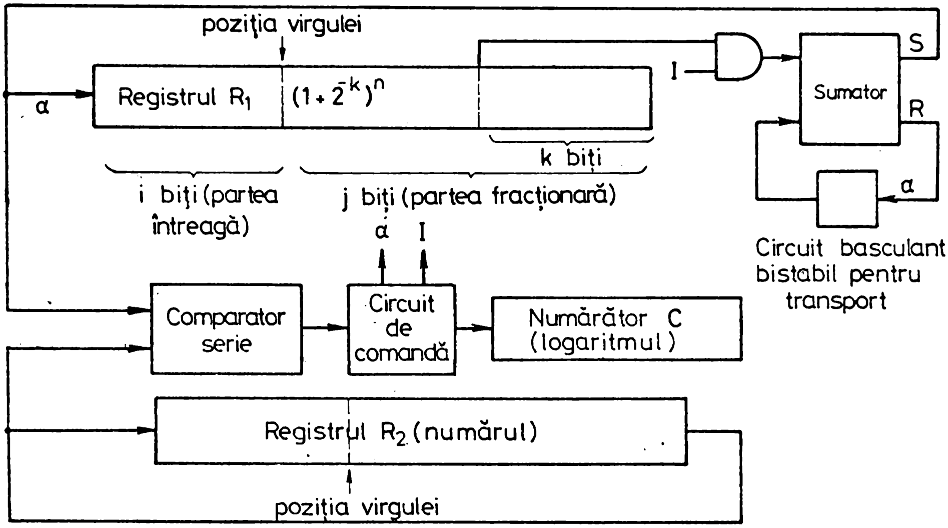


Fig.1-9. Dispozitivul de generare a logaritmului după algoritmul Nicaud.

Pentru obţinerea logaritmului binar, întrucît o bază u=2 nu asigură precizie, se utilizează o bază v de forma

$$v = 1 + \sum_{i=1}^{s} 2^{-k_i} \qquad (1\text{-}15)$$

care îndeplineşte condiţia

$$v^m = 2 \qquad (1\text{-}16)$$

m fiind un număr întreg egal cu cantitatea de cifre binare pe care trebuie să le aibă mantisa logaritmului binar. Inmulţirile cu o bază de forma (1-15) conduc la deplasări şi adunări a mai multor numere simultan [45] fiind deci necesar un sumator cu mai multe intrări în fig.1-9. Se trece apoi uşor de la logaritmul în baza v la logaritmul în baza 2.

Circuitele propuse pentru calculul logaritmurilor în baza $u = 1+2^{-k}$ sînt simple. Si în cazul acestei metode apar însă erori relative mari - de ordinul 1% - care includ eroarea de cuantificare (baza u nu se poate lua prea mică deoarece creşte atunci exagerat numărul de cicluri ale procesului iterativ) şi eroarea de rotunjire la înmulţire.

Avînd numere binare cu maximum 2o cifre întregi şi 13 cifre fracţionare, pentru a obţine o eroare de 1% la calculul logaritmului este utilizată o valoare k = 7. Cu o frecvenţă a impulsurilor de deplasare de 5 MHz rezultă un timp de calcul de

BUPT

15 ms pentru dispozitivul serie din fig.1-9 şi de cca 4 ms pentru un dispozitiv paralel.

Creşterea preciziei dispozitivului se poate face numai în contul creşterii timpului de calcul, prin adoptarea unei valori mari pentru exponentul k.

Performanţele reduse privind viteza şi precizia de calcul fac ca acest algoritm să fie utilizabil numai în instalaţii cu destinaţie specială.

1.2. Algoritmi de precizie ridicată

1.2.1. Al doilea algoritm Dean

In lucrarea [7] K.J.Dean descrie o posibilitate de sinteză a unui generator de logaritmi binari. Acesta constă dintr-un registru comandat, un registru de deplasare şi circuitele de comandă (fig.1-1o).

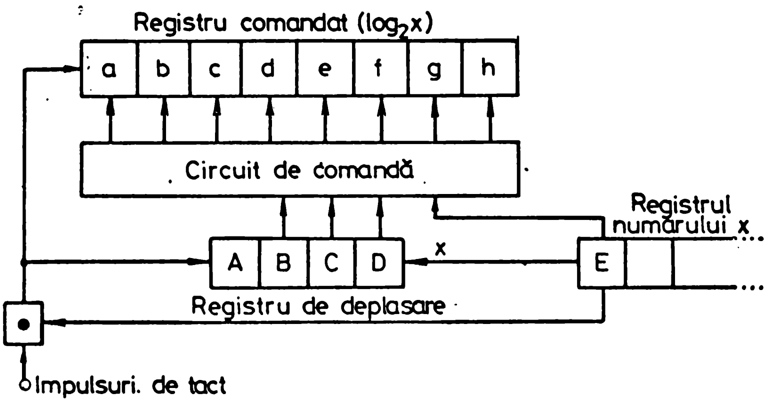


Fig.1-1o. Generatorul de logaritmi după al doilea algoritm Dean.

Algoritmul pleacă de la un tabel de adevăr (Tabelul 1-6) în care sînt trecute toate valorile numărului x cu patru biţi care au mantisa distinctă şi mantisele lui $\log_2 x$ calculate matematic avînd numai eroarea dată de rotunjirea la 8 biţi.

Iniţial registrul de deplasare este adus în starea oooo iar registrul comandat în starea 11111111 deoarece $\log_2 0 = \infty$. Numărul x este introdus în serie în registrul de deplasare cu bitul cel mai semnificativ în faţă. Circuitul de comandă examinează stările celulelor B,C,D,E ale registrului de deplasare şi a registrului în care este păstrat numărul x şi în funcţie de acestea stabileşte conţinutul corespunzător al registrului comandat (Tabelul 1-6). Se ţine cont că numerele de forma ooo1, oo1o, o1oo, 1ooo au aceeaşi mantisă.

Tabelul 1-6

x				mantisa lui $\log_2 x$							
A	B	C	D	a	b	c	d	e	f	g	h
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	1	0	1	1
1	0	1	0	0	1	0	1	0	0	1	0
1	0	1	1	0	1	1	1	0	1	0	1
1	1	0	0	1	0	0	1	0	1	0	1
1	1	0	1	1	0	1	1	0	0	1	1
1	1	1	0	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	0	1	0	0	0

Nu este indicată posibilitatea generării antilogaritmilor în [7] dar aceasta se poate rezolva printr-o metodă similară.

In [7] se prezintă modul de proiectare logică a circuitului de comandă pentru cazul cînd numărul x are 4 biţi. Sistemul poate fi dezvoltat la un număr oricît de mare de biţi şi poate asigura precizia dorită. Totuşi în acest caz proiectarea devine foarte greoaie din cauza necesităţii de a se minimiza funcţii logice cu un număr mare de variabile. Deasemenea, circuitul de comandă devine în acest caz deosebit de complex.

Timpul necesar generării logaritmului binar cu dispozitivul din fig.1-1o este datorat în cea mai mare parte introducerii în serie a numărului x în registrul de deplasare.

Dificultăţile arătate mai sus fac ca acest algoritm să nu fie aplicat deocamdată în calculatoare numerice universale sau calculatoare de birou. Utilizarea minimizării cu calculatorul a funcţiilor logice cu multe variabile şi utilizarea circuitelor integrate pe scară largă [26] pot permite în viitor realizarea unor astfel de generatoare de logaritmi binari cu un număr mare de biţi.

1.2.2. Al treilea algoritm Dean

In lucrarea [19] K.J.Dean prezintă un algoritm de calcul al logaritmilor şi antilogaritmilor binari precum şi schemele logice cu ajutorul cărora se realizează algoritmul. Schema pentru generarea logaritmilor binari utilizează o structură logică celulară de ridicare la patrat iar schema pentru generarea antilogaritmilor binari utilizează o structură logică celulară pentru

BUPT

extragerea rădăcinii patrate. Schemele sînt simple deşi cantitatea de circuite din structurile logice celulare este foarte mare. Ele pot asigura precizia dorită dacă utilizează un număr suficient de mare de cifre binare.

Algoritmul Dean este valabil pentru numere binare A cuprinse în domeniul :

$$1 \leqslant A < 2 \qquad (1\text{-}17)$$

Rezultă că algoritmul permite determinarea mantisei logaritmului oricărui număr N după ce caracteristica logaritmului a fost găsită cu algoritmul Mitchell [1]. Numărul A se obţine din numărul N prin stabilirea virgulei în dreapta bitului 1 cel mai semnificativ al acestuia.

Logaritmul numărului A se poate pune în forma :

$$\log_2 A = b_0 2^0 + b_1 2^{-1} + b_2 2^{-2} + \ldots + b_n 2^{-n} \qquad (1\text{-}18)$$

unde b_i reprezintă bitul de ordinul i al logaritmului iar n - numărul de biţi ai logaritmului. Relaţia (1-18) este pusă în forma:

$$\log_2 A = b_1 \log_2 2^{\frac{1}{2}} + b_2 \log_2 2^{\frac{1}{4}} + \ldots + b_n \log_2 2^{\frac{1}{2^n}} \qquad (1\text{-}19)$$

deoarece b_0 este întotdeauna nul cînd numărul A este cuprins în domeniul (1-17) şi deoarece :

$$\log_2 2^{\frac{1}{2^i}} = \frac{1}{2^i} = 2^{-i} \qquad (1\text{-}20)$$

Rezultă astfel din relaţia (1-19) prin antilogaritmare :

$$A = 2^{\frac{1}{2} b_1} \, 2^{\frac{1}{4} b_2} \ldots 2^{\frac{1}{2^n} b_n} \qquad (1\text{-}21)$$

In continuare algoritmul realizează determinarea succesivă a biţilor b_i ai logaritmului prin ridicări succesive la patrat şi împărţiri cu 2. Astfel, se efectuează operaţia :

$$A^2 = 2^{b_1} \, 2^{\frac{1}{2} b_2} \ldots 2^{\frac{1}{2^{n-1}} b_n} \qquad (1\text{-}22)$$

Dacă A^2 rezultă mai mare sau egal cu 2 (apare un 1 în poziţia 2^1 a numărului A^2) atunci $b_1 = 1$, se face o împărţire cu $2^{b_1} = 2$ a lui A^2 pentru a se înlătura factorul 2^{b_1} :

$$A_1 = \frac{A^2}{2} \qquad (1\text{-}23)$$

şi se repetă procedeul pentru determinarea bitului b_2 al logaritmului prin ridicarea la patrat a lui A_1.

BUPT

Dacă A^2 rezultă mai mic decît 2 (nu apare un 1 în poziția 2^1 a numărului A^2) atunci b_1 = o și nu se mai face împărțirea cu 2, deci :

$$A_1 = A^2 \qquad (1\text{-}24)$$

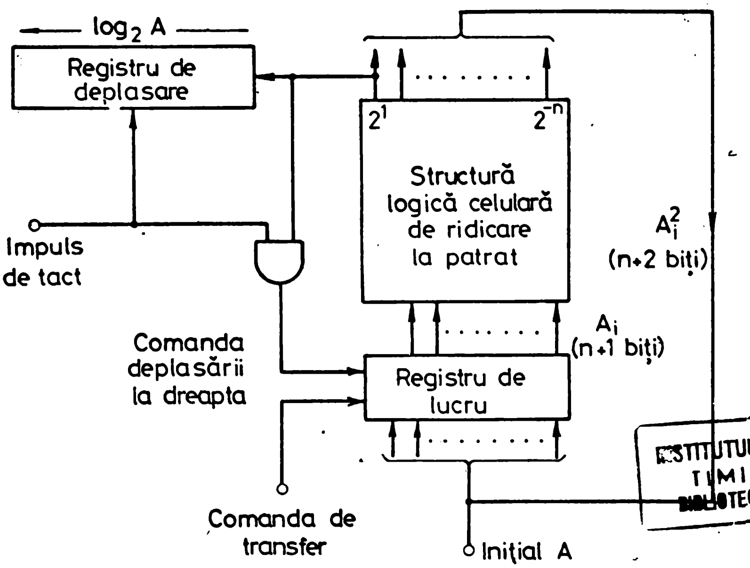
continuîndu-se cu o ridicare la patrat pentru determinarea lui b_2 [19].

Procedeul este continuat pînă la determinarea tuturor biților $b_1 \ldots b_n$ ai logaritmului binar. Procesul iterativ se desfășoară deci pe baza relațiilor :

$$\begin{cases} b_i = 1 & \text{dacă} \quad A_i^2 \geq 2 \\ b_i = 0 & \text{dacă} \quad A_i^2 < 2 \end{cases} \qquad (1\text{-}25)$$

$$\begin{cases} A_{i+1} = \dfrac{A_i^{\,2}}{2} & \text{dacă} \quad b_i = 1 \\ A_{i+1} = A_i^{\,2} & \text{dacă} \quad b_i = 0 \end{cases} \qquad (1\text{-}26)$$

In fig.1-11 se prezintă circuitul de generare a logaritmului binar bazat pe algoritmul de mai sus. El cuprinde un registru de lucru (ce conține inițial numărul A apoi, după ciclul i al iterației, numărul A_i), un registru de deplasare (în care se formează bit cu bit logaritmul) și o structură logică celulară de ridicare la patrat. Aceasta din urmă poate fi în particular o structură logică celulară de multiplicare.



INSTITUTUL POLITEHNIC TIMIȘOARA BIBLIOTECA CENTRALĂ

Fig.1-11. Dispozitivul de generare a logaritmilor după al treilea algoritm Dean.

BUPT

Numărul A_i este aplicat în paralel la intrările structurii logice celulare de ridicare la patrat. La ieşirea acesteia se obţine A_i^2 care se transferă în registrul de lucru. Dacă la ieşirea corespunzătoare poziţiei binare 2^1 a apărut un 1 atunci cu ajutorul unui impuls de tact se introduce bitul $b_i = 1$ în registrul de deplasare şi se comandă deplasarea la dreapta a registrului de lucru (împărţirea cu 2).

Pentru generarea antilogaritmilor în [19] K.J.Dean propune un procedeu asemănător, în care procesul iterativ este desfăşurat în sens contrar. Astfel ridicarea la patrat este înlocuită cu extragerea rădăcinii patrate iar împărţirea cu 2 este înlocuită printr-o înmulţire cu 2. Fiind dat logaritmul se pleacă de la numărul 1,000... care constituie antilogaritmul lui 0,000... şi se examinează bitul cel mai puţin semnificativ al logaritmului - b_n. Dacă b_n = 1 se dublează numărul iniţial şi se extrage rădăcina patrată. Dacă b_n = 0 se extrage rădăcina patrată din numărul iniţial fără dublarea lui. Procesul se continuă în acelaşi mod prin examinarea următorului bit al logaritmului. După extragerea rădăcinii patrate în urma examinării bitului b_1, în registrul de lucru se găseşte antilogaritmul.

Circuitul care generează antilogaritmul binar [19] este prezentat în fig.1-12 şi se deosebeşte de cel din fig.1-11 prin aceea că apare aici o structură logică celulară de extragere a rădăcinii patrate.

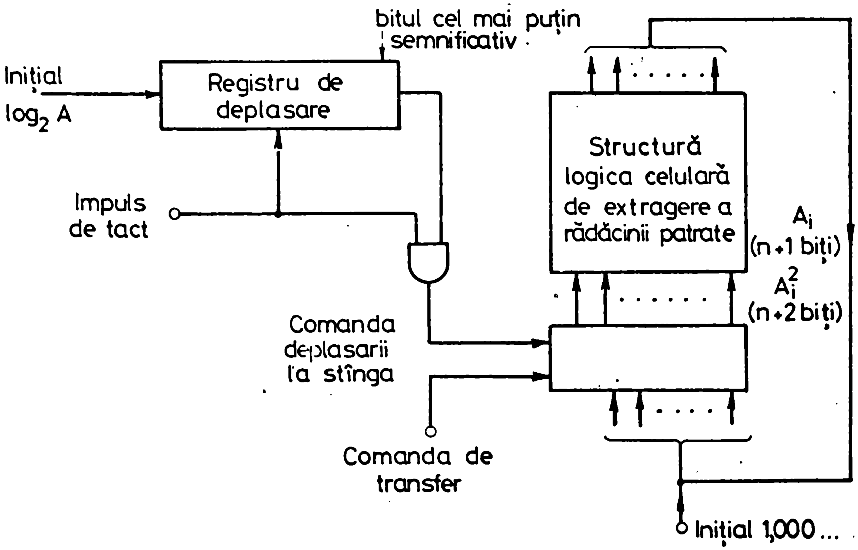


Fig.1-12. Dispozitivul de generare a antilogaritmilor după al treilea algoritm Dean.

BUPT

In calculul logaritmilor apare o eroare datorată rotunjirii rezultatului de lungime dublă de la ieşirea structurii logice celulare de ridicare la patrat şi datorită eliminării unui bit prin deplasarea la dreapta a conţinutului registrului de lucru.

La calculul antilogaritmului eroarea apare datorită determinării rădăcinii patrate cu un număr finit de biţi şi va fi mai mică decît la calculul logaritmului.

Valoarea erorii maxime ce apare în aceste calcule nu a fost determinată în [19] iar în lucrarea [20] care analizează eroarea metodei se arată că aceasta afectează bitul 21 al logaritmului dar nu se precizează cîţi biţi au numerele.

După studiul erorilor făcut de autorul tezei pentru un generator de antilogaritmi asemănător [56] se ajunge la concluzia că pentru n = 24 biţi eroarea afectează ultimii 4 biţi.

Durata operaţiilor de calcul al logaritmului sau antilogaritmului este determinată în primul rînd de cantitatea de biţi ai numerelor şi de timpul necesar propagării informaţiei prin structura logică celulară. Acest timp este relativ ridicat [22], [12], [23], [29], [30] iar el intră în durata totală a operaţiilor de n ori - n fiind cantitatea de biţi a mantisei logaritmului (timpul nu a fost analizat de autorul algoritmului).

Analizînd durata operaţiilor după acest algoritm, autorul tezei a dedus următoarele. Dacă se utilizează o structură logică celulară de multiplicare de tipul celei din [14] rezultă un timp total de calcul al logaritmului binar :

$$t_{log} \cong n.t_{rp} \qquad (1\text{-}27)$$

unde t_{rp} este timpul de ridicare la patrat :

$$t_{rp} = (n+1)t_1 + (2n+1)t_2 + t_3 \qquad (1\text{-}28)$$

cu : t_1 - timpul de propagare pe verticală într-o celulă din structura logică celulară, peste trei nivele logice,

t_2 - timpul de propagare pe orizontală într-o celulă a structurii (peste două nivele logice) iar

t_3 - timpul necesar în fiecare ciclu pentru deplasarea şi transferul conţinutului registrului de lucru. Considerînd timpul de propagare pe un nivel logic t_p rezultă $t_1 = 3\ t_p$, $t_2 = 2\ t_p$ şi deoarece t_3 este de ordinul 6 t_p se poate scrie :

$$t_{log} = n(7n+12)t_p \qquad (1\text{-}29)$$

Dacă se utilizează circuite integrate cu timp de propagare pe nivel $t_p = 6$ ns atunci pentru n=28 biţi rezultă $t_{log} = 35\ \mu s$.

BUPT

Ulterior [23] Dean a propus o structură logică celulară specială pentru ridicarea la patrat este de cîteva ori mai rapidă decît una de multiplicare [23].

Durata operaţiei de antilogaritmare este mai mare din cauza utilizării unei structuri logice celulare mai lente [12],[29],[3o]. In [12] nu sînt calculate numărul de celule N_c şi durata operaţiei de extragere a rădăcinii patrate t_{er} dar acestea se pot determina cu relaţiile de mai jos scrise în cazul cînd se determină rădăcina patrată cu n biţi şi se completează numărul din care se extrage rădăcina cu zerouri pînă la 2n biţi :

$$N_c = 2 + \frac{(n+6)(n-1)}{2} \qquad \text{(pentru n = par)} \qquad (1\text{-}3o)$$

$$t_{er} < t_1\left[N_c - 2(n-1)\right] + t_2 N_c + t_3 \qquad (1\text{-}31)$$

$$t_{alog} = n.t_{er} \qquad (1\text{-}32)$$

unde : $t_1 = 2\ t_p$ este timpul de propagare pe orizontală spre stînga într-o celulă [12], $t_2 = t_p$ reprezintă timpul de propagare pe orizontală spre dreapta într-o celulă, iar $t_3 = 6\ t_p$ este timpul de deplasare şi transferul conţinutului registrului de lucru. Astfel, pentru n = 28 şi t_p = 6 ns rezultă N_c = 461 şi $t_{alog} \cong 214,6\ \mu s$.

Schemele propuse în [19] reprezintă scheme de tip serie-paralel şi durata calculului logaritmului sau antilogaritmului este în consecinţă ridicată. Deasemenea, cantitatea mare de circuite utilizate - comparabilă cu cea a unor structuri logice celulare de înmulţire şi împărţire mult mai rapide [11],[14]- face ca algoritmul şi schemele propuse de Dean [19] să nu poată fi folosite în calculatoare electronice în scopul efectuării înmulţirii şi împărţirii cu ajutorul logaritmilor ci numai în scopul calculării logaritmilor pentru operaţii mai complicate cablate (ridicarea la o putere fracţionară, extragerea rădăcinii de orice ordin, calculul logaritmilor în alte baze).

Algoritmul Dean îşi poate găsi o aplicaţie în calculatoare electronice de birou [2o], unde viteza nu este esenţială şi nu se pretind rezultate întregi. Deasemenea el este aplicabil în instalaţii de calcul specializate în care generarea logaritmilor şi antilogaritmilor cu mare precizie este indispensabilă. Precizia algoritmului se măreşte simplu, prin creşterea numărului de biţi n şi deci extinderea simplă a circuitelor electronice.

In lucrarea [37] autorul tezei a propus deasemenea o structură logică celulară de ridicare la patrat, destinată algoritmului de mai sus, mai simplă şi mai rapidă decît o structură logică celulară de multiplicare.

BUPT

1.2.3. Algoritmul Perle

In lucrarea [24] se prezintă în formă generală un algoritm pentru obținerea logaritmului și antilogaritmului în orice bază printr-un calcul iterativ.

Astfel, dacă este dat un număr oarecare N_g și trebuie calculat $\log_b N_g$ atunci se deduce cel mai mare număr b^m (m fiind un întreg pozitiv) care îndeplinește condiția :

$$N_o = b^m < N_g \tag{1-33}$$

In acest caz m este chiar caracteristica logaritmului și se poate arăta că :

$$m < \log_b N_g \leqslant m+1 \tag{1-34}$$

Algoritmul adună apoi o serie de numere la cantitatea inițială $N_o = b^m$ pentru ca aceasta să se apropie de N_g. Seria de numere se adoptă astfel încît să satisfacă două condiții :

- să permită aproximarea numărului N_g pînă la orice eroare dată,
- să permită calculul simplu al logaritmului noului număr obținut după fiecare adunare.

Se propune deci operația iterativă :

$$N_i = N_{i-1} + N_{i-1} \cdot 2^{-i} \quad \text{pentru } i = 1...M \tag{1-35}$$

care, după autor [24], prezintă o bună convergență. Dacă în urma unei iterații rezultă $N_i > N_g$ atunci iterația corespunzătoare este anulată și se mărește cu 1 valoarea lui i.

Din relația (1-35) se obține prin logaritmare :

$$\log_2 N_i = \log_b N_{i-1} + \log_b (1+2^{-i}) \tag{1-36}$$

și deci, pentru calculul acestui logaritm, trebuie cunoscut setul de valori :

$$\log_b (1+2^{-i}), \quad \text{pentru } i=1...M \tag{1-37}$$

In cazul operației de antilogaritmare, fiind dată cantitatea $\log_b N_g$, avînd o parte întreagă egală cu m și o parte fracționară Δ_o se urmărește calcularea lui N_g. Plecînd de la valorile inițiale

$$N_o = b^m \tag{1-38}$$

și Δ_o, se efectuează iterațiile :

$$\Delta_i = \Delta_{i-1} - \log_b (1+2^{-i}) \quad \text{pentru } i= 1...M \tag{1-39}$$

prin care Δ_i tinde către zero și :

$$N_i = N_{i-1} + N_{i-1} \cdot 2^{-i} \tag{1-40}$$

Dacă în urma unei iterații rezultă $\Delta_i < 0$ iterația cores-

BUPT

punzătoare este anulată şi se măreşte cu 1 valoarea lui i.

Relaţiile (1-35),(1-36),(1-39) şi (1-4o) arată că se poate efectua atît calculul logaritmului cît şi al antilogaritmului prin operaţii simple de adunare şi deplasare pentru care se poate concepe o schemă comună, cu deosebirea că la calculul antilogaritmului trebuie efectuată o scădere (relaţia 1-39).

Algoritmul Perle [24] este cel mai semnificativ algoritm de calcul al logaritmilor şi antilogaritmilor din cele prezentate în acest capitol. El poate fi aplicat atît la generarea prin hardware cît şi la calcule prin program asigurîndu-se o precizie dorită prin adoptatea unei cantităţi corespunzătoare pentru partea întreagă şi fracţionară a numerelor. Precizia este totuşi limitată în cazul calculului prin program, dacă se lucrează cu lungime simplă de cuvînt, de numărul biţilor cuvîntului din calculator.

Autorul algoritmului nu prezintă propuneri concrete pentru circuitele de logaritmare-antilogaritmare şi deci nu se poate stabili durata operaţiilor.

Se poate totuşi aprecia că durata maximă a operaţiilor de logaritmare şi antilogaritmare realizate într-un dispozitiv aritmetic cu virgulă mobilă rapid (cu sumator cu transport simultan) ce operează cu numere avînd n biţi, dacă nu se consideră timpul necesar obţinerii caracteristicii logaritmului rezultă de ordinul:

$$t_{log\ max} \leq 3\ n\ t_t \qquad (1\text{-}41)$$

unde t_t este perioada de tact a dispozitivului. Astfel, pentru $n = 28$ biţi şi $t_t = 25o$ ns rezultă

$$t_{log\ max} = 21\ \mu s \qquad (1\text{-}42)$$

care este comparabilă cu durata unei înmulţiri sau împărţiri dintr-un dispozitiv aritmetic cu virgulă mobilă [74]. Deci implementat pe un dispozitiv aritmetic clasic algoritmul Perle nu poate asigura reducerea duratei înmulţirii sau împărţirii în cazul efectuării operaţiilor cu ajutorul logaritmilor.

Deasemenea, determinarea caracteristicii logaritmului în altă bază decît 2, cînd numerele sînt reprezentate în sistemul binar, constituie o operaţie dificilă.

Pentru obţinerea unei erori oricît de mici autorul preconizează creşterea numărului de iteraţii. Cînd se lucrează însă cu numere (deci şi cu circuite) avînd cantitatea de cifre binare n stabilită continuarea iteraţiei pentru valori $i > n$ nu are sens deoarece prin deplasarea la dreapta cu i poziţii a rezultatului anterior, (1-36 şi 1-4o) cantităţile ce trebuie adunate devin mai mici decît 2^{-n} şi se află în afara domeniului adoptat pentru numere. Problema

BUPT

erorii calculului trebuie deci analizată plecînd de la o cantitate de cifre binare (de iteraţii) n dată (cum s-a procedat în /58/).

Algoritmul Perle mai prezintă dezavantajul că în cadrul operaţiilor descrise de relaţiile (1-35),(1-36) şi (1-4o) este permanent prezentă partea întreagă a numărului ce se logaritmează, respectiv caracteristica logaritmului, ceea ce măreşte timpul de calcul şi complexitatea circuitelor. Cum s-a arătat în /58/ în iteraţii se pot efectua operaţii numai asupra părţii fracţionare a numărului sau mantisei logaritmului.

Un alt dezavantaj al algoritmului Perle îl constituie prezenţa unor operaţii aritmetice diferite (adunare şi scădere) în relaţiile (1-36) şi (1-39) ceea ce complică suplimentar circuitele logice.

Pe de altă parte scrierea relaţiilor (1-33) şi (1-34) în forma :

$$b^m \leqslant N_g \qquad (1\text{-}33a)$$

$$m \leqslant \log_n N_g < m+1 \qquad (1\text{-}34a)$$

ar permite reducerea numărului de iteraţii şi deci a timpului în cazul îndeplinirii condiţiei de egalitate.

Lucrarea lui D.Perle [24] are însă şi o scăpare pe care autorul tezei a sesizat-o în lucrarea /72/. Si anume, în unele situaţii iteraţia propusă de Perle nu este convergentă. Astfel, dacă trebuie găsit logaritmul unui număr N_g foarte apropiat de b^{m+1}, chiar adunînd în iteraţie toţi termenii $N_{i-1}.2^{-i}$, pentru i = 1...M, rezultă :

$$N_M = N_o \prod_{i=1}^{i=M} (1+2^{-i}) \cong 2{,}38\ b^m \qquad (1\text{-}43)$$

care faţă de $N_g \cong b^{m+1}$ este, pentru $b > 2{,}38$:

$$2{,}38\ b^m < b^{m+1} \qquad (1\text{-}44)$$

Deci numărul N_g nu poate fi aproximat decît în cazul cînd b = 2 prin algoritmul Perle.

Pentru b> 2 iteraţia propusă de Perle este posibilă numai dacă o serie de etape ale iteraţiei se repetă. Cel mai redus număr de repetări, în situaţia cînd $N_g \cong b^{m+1}$, se obţine dacă se repetă numai iteraţia de ordinul i=1 de atîtea ori încît să nu mai fie necesară repetarea altora.

Se poate arăta de cîte ori trebuie repetată în calcul iteraţia de ordinul i=1 scriind condiţia :

$$b^m(1+2^{-1})^x \cong b^{m+1} \qquad (1\text{-}45)$$

de unde :

BUPT

$$x \cong \frac{\log_{10} b}{\log_{10}(1+2^{-1})} \cong \frac{\log_{10} b}{0{,}176} \cong 5{,}7 \log_{10} b \qquad (1\text{-}46)$$

In tabelul 1-7 se dau valorile lui x pentru diferite base b. Iterația de ordinul i=1 trebuie deci repetată de cel mult atîtea ori cît arată partea întreagă a lui x. Partea fracționară a lui x este acoperită în iterație de prezența termenilor de ordin $i>1$. Autorul tezei a arătat în /58/ că un termen de ordin i+1 nu poate să apară de două ori în iterație deoarece:

Tabelul 1-7

b	x fracționar	x întreg
2	1,71	1
4	3,42	3
8	5,13	5
1o	5,7o	5
16	6,85	6

$$\left[1+2^{-(i+1)}\right]^2 > (1+2^{-i}) \qquad (1\text{-}47)$$

ceea ce înseamnă că o iterație dublă de ordinul i+1 se poate înlocui cu o iterație simplă de ordinul i.

Astfel algoritmul Perle prezintă desavantajul că pentru o bază $b>2$ este necesară repetarea de cîteva ori a primei iterații, fapt sesizat de autorul tezei în /72/.

O problemă asemănătoare apare și la calculul antilogaritmului cu algoritmul Perle.

1.2.4. Algoritmul Schmid

In lucrarea [65] se prezintă un algoritm de calcul al logaritmilor naturali ai numerelor reprezentate în sistemul zecimal, utilizabil în calculatoarele electronice de birou, care poate asigura o precizie oricît de bună.

Astfel, scriind :

$$\ln\left(x \prod_{i=o}^{i=n} a_i\right) = \ln x + \sum_{i=o}^{i=n} \ln a_i \qquad (1\text{-}48)$$

unde constantele a_i se aleg astfel încît produsul $\prod_{i=o}^{i=n}$ să tindă rapid spre 1, se obține :

$$\ln x \cong - \sum_{i=1}^{i=n} \ln a_i \qquad (1\text{-}49)$$

Operația reprezintă o iterație în care factorii a_i se repetă de atîtea ori cît este necesar iar valoarea acestora se adoptă :

$$a_i = 1 + (-1)^i 1o^{-i} \qquad (1\text{-}5o)$$

ceea ce face ca înmulțirea să se transforme într-o adunare sau scădere (ca și în [2],[3],[24],/58/).

Acest algoritm poate fi aplicat și la calculul logaritmilor binari. Totuși el nu este aplicabil și la calculul antilogaritmilor cu aceleași avantaje deoarece ar fi necesară o cantitate

BUPT

mare de împărţiri, care nu se mai pot reduce la simple adunări sau scăderi.

Alte dezavantaje ale acestui algoritm le constituie :

- repetarea de mai multe ori a iteraţiei cu aceeaşi constantă a_1, ceea ce duce la creşterea duratei operaţiei,
- folosirea logaritmilor negativi a căror valoare este cuprinsă într-un domeniu larg, ceea ce duce la micşorarea preciziei de calcul şi la folosirea ineficientă a circuitelor electronice pentru anumite valori ale operanzilor.
- necesitatea a două tipuri de operaţii : adunare şi scădere.

...

BUPT

CAPITOLUL II

NOI ALGORITMI PENTRU CALCULUL LOGARITMILOR ŞI ANTILOGARITMILOR BINARI.

2.1. Algoritm bazat pe înmulţirea şi împărţirea cu constante a numărului ce se logaritmează sau antilogaritmează.

In lucrarea [19] se prezintă al treilea algoritm Dean de calcul al logaritmilor şi antilogaritmilor binari precum şi circuitele logice cu ajutorul cărora se poate realiza algoritmul. Circuitul pentru generarea logaritmilor binari utilizează o structură logică celulară pentru ridicare la patrat iar circuitul pentru generarea antilogaritmilor binari utilizează o structură logică celulară pentru extragerea rădăcinii patrate.

Este însă posibil ca într-o unitate aritmetică sau echipament de calcul să existe deja un dispozitiv de înmulţire şi împărţire obişnuit sau celular. In acest caz, folosind algoritmul de mai sus, elaborat de autorul tezei [66], se pot calcula logaritmii şi antilogaritmii binari ai numerelor binare numai prin înmulţiri şi împărţiri ceea ce face să nu mai fie necesare circuite pentru ridicare la patrat (aceasta se poate face de altfel prin înmulţire) şi pentru extragerea rădăcinii patrate.

Algoritmul este valabil este valabil pentru numere binare cuprinse în domeniul

$$1 \leq A < 2 \qquad (2\text{-}1)$$

în care se poate aduce orice număr prin deplasări sau normalizare cînd se determină caracteristica logaritmului [1].

Logaritmul unui astfel de număr se poate scrie în forma :

$$\log_2 A = b_0 2^0 + b_1 2^{-1} + b_2 2^{-2} + \ldots + b_i 2^{-i} \qquad (2\text{-}2)$$

unde b_i reprezintă bitul de ordinul i al logaritmului binar. După cum se arată în lucrarea [19] expresia (2-2) se poate pune în forma:

$$\log_2 A = b_1 \log_2 2^{\frac{1}{2}} + b_2 \log_2 2^{\frac{1}{4}} + \ldots + b_i \log_2 2^{\frac{1}{2^i}} \qquad (2\text{-}3)$$

deoarece b_0 este întotdeauna zero cînd numărul A este cuprins în domeniul (2-1) şi deoarece :

$$\log_2 2^{\frac{1}{2^i}} = \frac{1}{2^i} = 2^{-i} \qquad (2\text{-}4)$$

Rezultă astfel relaţia cunoscută [19] :

BUPT

Pentru determinarea succesivă a biţilor b_i ai logaritmului în [19] se efectuează ridicări succesive la patrat şi împărţiri cu 2 (deplasări) aşa cum s-a arătat în Capitolul I al tezei. Calculul antilogaritmului se face în [19] printr-un procedeu asemănător parcurgînd etapele în sens contrar. Astfel, operaţia de ridicare la patrat este înlocuită cu extragerea rădăcinii patrate iar împărţirea cu 2 - printr-o înmulţire cu 2.

Algoritmul propus de autorul tezei /66/ constă în următoarele. Plecînd de la expresia (2-5) se poate determina bitul b_1 printr-o înmulţire cu $2^{\frac{1}{2}}$:

$$A_1 = 2^{\frac{1}{2}} A = 2^{\frac{b_1+1}{2}} \, 2^{\frac{1}{4} b_2} \, 2^{\frac{1}{8} b_3} \ldots 2^{\frac{1}{2^i} b_0} \qquad (2\text{-}6)$$

Astfel: a. dacă $A_1 \geq 2$ (apare un 1 în poziţia 2^1 a numărului A_1 cauzat de primul factor al relaţiei (2-6)) rezultă $b_1=1$,

b. dacă $A_1 < 2$ (nu apare 1 în poziţia 2^1 a numărului A_1) rezultă $b_1=o$.

Pentru determinarea lui b_2 se continuă prin înmulţirea cu un alt factor şi anume :

a. dacă $b_1=1$ se împarte A_1 cu 2 pentru a se înlătura factorul $2^{\frac{b_1+1}{2}}$ şi se înmulţeşte noua cantitate cu $2^{\frac{3}{4}}$:

$$A_2 = \frac{1}{2} \, 2^{\frac{3}{4}} A_1 = 2^{-\frac{1}{4}} A_1 = \frac{A_1}{2^{\frac{1}{4}}} \qquad (2\text{-}7)$$

Deci trebuie făcută o împărţire cu numărul $2^{\frac{1}{4}}$ şi rezultă :

$$A_2 = 2^{\frac{3+b_2}{4}} \, 2^{\frac{1}{8} b_3} \ldots 2^{\frac{1}{2^i} b_i} \qquad (2\text{-}8)$$

b. dacă $b_1=o$ nu mai este necesară împărţirea cu 2 şi se efectuează înmulţirea :

$$A_2 = 2^{\frac{1}{4}} A_1 \qquad (2\text{-}9)$$

Rezultă astfel :

$$A_2 = 2^{\frac{1}{4}} A_1 = 2^{\frac{1}{4}} \, 2^{\frac{1}{2}} \, 2^{\frac{1}{4} b_2} \, 2^{\frac{1}{8} b_3} \ldots 2^{\frac{1}{2^i} b_i} =$$

$$= 2^{\frac{3+b_2}{4}} \, 2^{\frac{1}{8} b_3} \ldots 2^{\frac{1}{2^i} b_i} \qquad (2\text{-}1o)$$

(pentru că $b_1=o$) adică aceeaşi expresie ca şi în cazul a, (2-8).

Apare acum una din situaţiile :

BUPT

a. $A_2 \geqslant 2$ deci $b_2 = 1$, datorită primului factor din relaţia (2-1o) sau

b. $A_2 < 2$ deci $b_2 = 0$.

Pentru determinarea lui b_3 se efectuează operaţiile :

a. dacă $b_2 = 1$:

$$A_3 = \frac{1}{2}\, 2^{\frac{7}{8}} A_2 = 2^{-\frac{1}{8}} A_2 = \frac{A_2}{2^{\frac{1}{8}}} = 2^{\frac{7+b_3}{8}}\, 2^{\frac{1}{16}b_4} \dots 2^{\frac{1}{2^i}b_i} \qquad (2\text{-}11)$$

b. dacă $b_2 = 0$:

$$A_3 = 2^{\frac{1}{8}} A_2 = 2^{\frac{1}{8}}\, 2^{\frac{3}{4}}\, 2^{\frac{1}{8}b_3}\, 2^{\frac{1}{16}b_4} \dots 2^{\frac{1}{2^i}b_i} = $$

$$= 2^{\frac{7+b_3}{8}}\, 2^{\frac{1}{16}b_4} \dots 2^{\frac{1}{2^i}b_i} \qquad (2\text{-}12)$$

şi apare una din situaţiile :

a. $A_3 \geqslant 2$ deci $b_3 = 1$ sau

b. $A_3 < 2$ deci $b_3 = 0$

In acest fel operaţia care se face la un moment dat depinde de valoarea bitului logaritmului determinat în pasul anterior. In general, pentru determinarea bitului b_i se procedează astfel :

a. dacă $b_{i-1} = 1$ se face împărţirea :

$$A_i = \frac{A_{i-1}}{\frac{1}{2^{2^i}}} = \frac{A_{i-1}}{C_i} \qquad (2\text{-}13)$$

b. dacă $b_{i-1} = 0$ se face înmulţirea :

$$A_i = 2^{\frac{1}{2^i}} A_{i-1} = C_i \cdot A_{i-1} \qquad (2\text{-}14)$$

şi rezultă una din situaţiile :

a. A_i 2 deci $b_i = 1$ sau

b. A_i 2 deci $b_i = 0$

Trebuie remarcat că întotdeauna $b_0 = 0$ iar prima operaţie la logaritmare este o înmulţire, operaţie care se încadrează în algoritmul prezentat.

In loc de împărţirea cu $C_i = 2^{\frac{1}{2^i}}$ se poate face o împărţire cu 2 (o deplasare la dreapta) şi o înmulţire cu factorul $2^{\frac{2^i-1}{2^i}}$.

Din motivul că este mai simplu să se utilizeze întotdeauna acelaşi factor $\frac{1}{2^{2^i}}$ este indicat să se efectuese două

operaţii diferite - înmulţire şi împărţire. Uneori, pentru a evita împărţirea, poate să intereseze şi varianta în care se efectuează înmulţiri cu factori diferiţi :

$$2^{\frac{1}{2^i}} \quad \text{sau} \quad 2^{\frac{2^i-1}{2^i}}$$

Factorii de forma $2^{\frac{1}{2^i}}$ se notează în continuare cu C_i. Ei reprezintă constante care au fost calculate de autorul tezei cu 43 cifre binare /73/ şi sînt prezentate în Tabelul 2-1.

TABELUL 2-1 C_i în zecimal şi în binar

1	1,4142135623730	1,0110101000001001111001100110011111110011101
2	1,1892071150027	1,0011000001101111111000001010001100011011011
3	1,0905077326652	1,0001011100101011100000111100011111010101000
4	1,0442737824274	1,0000101101010101100001101100111110011000100
5	1,0218971486541	1,0000010110011011000011010011000101011000010
6	1,0108892860516	1,0000001011001001101000111110011101111000000
7	1,0054299011128	1,0000000101100011110110101001111110110011001
8	1,0027112750502	1,0000000010110001101011111010010110101011110
9	1,0013547198921	1,0000000001011000110010000110110110100001110
10	1,0006771306930	1,0000000000101100011000000101111000101110100
11	1,0003385080526	1,0000000000010110001011110011100100000100000
12	1,0001692397053	1,0000000000001011000101110101111011111111110
13	1,0000846162726	1,0000000000000101100010111010000000011111101
14	1,0000423072413	1,0000000000000010110001011100110000110111110
15	1,0000211533969	1,0000000000000001011000101110010100100101111
16	1,0000105766425	1,0000000000000000101100010111001001010101011
17	1,0000052883072	1,0000000000000000010110001011100100011011010
18	1,0000026441501	1,0000000000000000001011000101110010001001110
19	1,0000013220742	1,0000000000000000000101100010111001000011111
20	1,0000006610368	1,0000000000000000000010110001011100100001101
21	1,0000003305183	1,0000000000000000000001011000101110010000110
22	1,0000001652591	1,0000000000000000000000101100010111001000011
23	1,0000000826295	1,0000000000000000000000010110001011100100001
24	1,0000000413147	1,0000000000000000000000001011000101110010000
25	1,0000000206573	1,0000000000000000000000000101100010111001000
26	1,0000000103286	1,0000000000000000000000000010110001011100100
27	1,0000000051643	1,0000000000000000000000000001011000101110010
28	1,0000000025821	1,0000000000000000000000000000101100010111001
29	1,0000000012910	1,0000000000000000000000000000010110001011100
30	1,0000000006455	1,0000000000000000000000000000001011000101110
31	1,0000000003227	1,0000000000000000000000000000000101100010111
32	1,0000000001613	1,0000000000000000000000000000000010110001011
33	1,0000000000806	1,0000000000000000000000000000000001011000101
34	1,0000000000403	1,0000000000000000000000000000000000101100010
35	1,0000000000201	1,0000000000000000000000000000000000010110001
36	1,0000000000100	1,0000000000000000000000000000000000001011000
37	1,0000000000050	1,0000000000000000000000000000000000000101100
38	1,0000000000025	1,0000000000000000000000000000000000000010110
39	1,0000000000012	1,0000000000000000000000000000000000000001011
40	1,0000000000006	1,0000000000000000000000000000000000000000101
41	1,0000000000003	1,0000000000000000000000000000000000000000010
42	1,0000000000001	1,0000000000000000000000000000000000000000001
43	1,0000000000000	1,0000000000000000000000000000000000000000000

Ele sînt cuprinse în domeniul :

$$1 < 2^{\frac{1}{2^i}} \leq \sqrt{2} \qquad (2\text{-}15)$$

şi tind, pentru $i \Rightarrow \infty$, spre valoarea 1.

Operaţia de antilogaritmare este descrisă de relaţia (2-5) adică se fac numai operaţii de înmulţire cu factori de forma $2^{\frac{1}{2^i}b_i}$ aşa cum autorul tezei a arătat şi în lucrarea /56/. Astfel, cînd bitul b_i al logaritmului este egal cu 1 se face înmulţirea cu factorul $2^{\frac{1}{2^i}}$ iar cînd bitul b_i este egal cu 0 nu se mai face înmulţirea deoarece factorul are valoarea 1.

La calculul logaritmului şi antilogaritmului binar pe baza algoritmului propus apar erori din următoarele motive :

- numărul limitat de cifre ale factorilor C_i ,
- rotunjirea rezultatului la n cifre după virgulă în urma fiecărei operaţii de înmulţire şi împărţire.

In cazul operaţiilor de împărţire eroarea datorată numărului finit de cifre ale factorilor va fi pozitivă şi ea va compensa parţial eroarea de rotunjire. In concluzie eroarea maximă apare numai în cazul în care se efectuează doar operaţii de înmulţire în fiecare pas al iteraţiei. Aceasta are loc la calculul logaritmului atunci cînd toate cifrele b_i ale logaritmului sînt egale cu 0 iar la calculul antilogaritmului atunci cînd cifrele b_i ale logaritmului sînt egale cu 1. Eroarea absolută maximă este aceeaşi în ambele cazuri. Ea se va calcula mai jos.

Relaţia (2-5) se poate transforma folosind substituţia :

$$(C_i)^{b_i} = 1+b_i(C_i-1) \qquad (2\text{-}16)$$

valabilă pentru valoarea 0 sau 1 a bitului b_i. Astfel :

$$A = \prod_{i=1}^{i=n} \left[1+b_i(C_i-1)\right] \qquad (2\text{-}17)$$

Dacă se notează produsul parţial obţinut după i iteraţii cu P_i atunci se poate scrie :

$$P_i = P_{i-1}\left[1+b_i(C_i-1)\right] = P_{i-1}+P_{i-1}b_i(C_i-1) \qquad (2\text{-}18)$$

In realitate, din cauza rotunjirii produsului după fiecare iteraţie, se face operaţia :

$$P_i = P'_{i-1} + P_{i-1}(C_i-1) \qquad (2\text{-}19)$$

unde P'_{i-1} reprezintă valoarea rotunjită la n biţi după virgulă a produsului P_{i-1} disponibil cu 2n biţi după virgulă şi s-a luat $b_i=1$.

BUPT

Pentru ca erorile să lipsească ar trebui efectuată operaţia:

$$(P_i+\Delta P_i) = (P_{i-1}+\Delta P_{i-1})+(P'_{i-1}+\Delta P_{i-1}+e_r)(C_i+e_r-1) \qquad (2\text{-}20)$$

unde e_r reprezintă eroarea de rotunjire şi eroarea datorată numărului finit de cifre al constantelor C_i.

Scăzînd din (2-20) pe (2-19) se obţine o relaţie de recurenţă care dă eroarea absolută în cadrul unei înmulţiri :

$$\Delta P_i = \Delta P_{i-1}+(\Delta P_{i-1}+e_r)(C_i-1)+(P'_{i-1}+\Delta P_{i-1}+e_r)e_r \qquad (2\text{-}21)$$

In această relaţie se pot particulariza următoarele mărimi:

$$e_r<2^{n}\,,\quad C_i-1<2^{-i}$$
$$P'_{i-1}+\Delta P_{i-1}+e_r< 2^{1} \qquad (2\text{-}22)$$

Cu aceasta :

$$\Delta P_i<\Delta P_{i-1}+(\Delta P_{i-1}+2^{-n})2^{-i}+2^{-(n-1)}=\Delta P_{i-1}(1+2^{-i})+2^{-n}(2^{-i}+2^{-1}) \qquad (2\text{-}23)$$

Dacă se înlocuieşte în paranteza a doua 2^{-1} cu $2^{0}=1$ atunci :

$$\Delta P_i<\Delta P_{i-1}(1+2^{-i})+2^{-n}(1+2^{-i}) = (\Delta P_{i-1}+2^{-n})(1+2^{-i}) \qquad (2\text{-}24)$$

Aplicînd această formulă de recurenţă pentru $i=2,3,\ldots,n$ rezultă în final eroarea absolută :

$$\Delta P_n< 2^{-n}\left[\prod_{i=2}^{i=n}(1+2^{-i})+\prod_{i=3}^{i=n}(1+2^{-i})+\ldots+\prod_{i=n}^{i=n}(1+2^{-i})\right] =$$

$$= 2^{-n}\sum_{j=2}^{j=n}\prod_{i=j}^{i=n}(1+2^{-i}) \qquad (2\text{-}25)$$

Cu această relaţie se poate delimita uşor eroarea absolută maximă a logaritmului sau antilogaritmului binar pentru un n dat.

In realitate eroarea de rotunjire e_r are valoarea medie de ordinul $2^{-(n+1)}$ deci de 2 ori mai mică decît s-a admis. Astfel eroarea absolută posibilă este de două ori mai mică decît limita superioară dată de relaţia (2-25) :

$$P_{n.p.} \cong 2^{-(n+1)}\sum_{j=2}^{j=n}\prod_{i=j}^{i=n}(1+2^{-i}) \qquad (2\text{-}26)$$

Astfel, pentru n = 25 rezultă :

$$P_{n.p.} \cong 2^{-26}.25{,}2<2^{-26}(2^{4}+2^{3}+2^{1}) = 2^{-22}+2^{-23}+2^{-25}\,, \qquad (2\text{-}27)$$

ceea ce înseamnă că eroarea afectează ultimele poziţii binare ale logaritmului sau antilogaritmului.

Eroarea absolută ce apare în calcule folosind acest algoritm are acelaşi ordin de mărime cu cea a algoritmului Dean [19].

BUPT

Deoarece eroarea absolută la calculul logaritmului şi antilogaritmului este întotdeauna negativă există posibilitatea de a se reduce eroarea absolută posibilă (relaţia 2-26) la jumătate prin adunarea unei corecţii egală cu $1/2\ \Delta P_{a.p.}$ la logaritm sau antilogaritm. Prin aceasta eroarea absolută poate fi pozitivă sau negativă iar în calculele cu logaritmii pot să apară unele compensări ale erorilor.

Algoritmul propus mai sus se poate utiliza atît pentru generarea logaritmilor şi antilogaritmilor binari prin hardware cît şi pentru calculul acestora cu subrutină în calculatoare electronice universale. In acest ultim caz nu se obţin însă avantaje faţă de algoritmele bazate pe dezvoltarea în serie rapid convergente utilizate în prezent. Folosit la generarea logaritmilor şi antilogaritmilor prin hardware algoritmul aduce avantajul vitezei mai mari faţă de alţi algoritmi (Capitolul III).

2.2. Algoritm bazat pe descompunerea în factori a numărului ce se logaritmează.

In /37/ şi /58/ autorul tezei a prezentat un algoritm de calcul al logaritmilor şi antilogaritmilor binari bazat pe descompunerea în anumiţi factori a numărului ce se logaritmează. Algoritmul poate asigura precizia dorită prin adoptarea unei cantităţi corespunzătoare de biţi pentru factori.

In [3] G.G.Ginkin a prezentat o posibilitate de obţinere a logaritmilor zecimali cu un număr mare de cifre prin descompunerea numărului ce se logaritmează în factori şi apoi adunarea logaritmilor factorilor daţi în tabele. Acest procedeu a fost adaptat de autorul tezei la sistemul binar de numeraţie /37/,/58/. Autorul tezei a evidenţiat particularităţile noului algoritm, precizia ce poate fi asigurată, posibilităţile de aplicare la generarea prin hardware a logaritmilor şi antilogaritmilor binari.

Similar cu descompunerea unui număr zecimal în factori [3] un număr binar N se poate descompune în factori de forma :

$$\begin{aligned} N = N_0(1+k_{11}2^{-1})\,(1+k_{12}2^{-1})\ldots \\ \ldots(1+k_{21}2^{-2})\,(1+k_{22}2^{-2})\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots(1+k_{n1}2^{-n})\,(1+k_{n2}2^{-n})\ldots \end{aligned} \qquad (2\text{-}28)$$

unde : - N_0 este numărul întreg cel mai apropiat de N (mai mic sau egal cu acesta),

- k_{ij} este "coeficient de creştere" egal cu 1 sau 0, care arată dacă factorul respectiv este sau nu prezent în

BUPT

descompunerea în factori (i=1...n)

- n este numărul de biţi ai factorilor, care poate fi egal sau mai mare decît cantitatea de biţi ai numărului N, în funcţie de precizia dorită).

In această descompunere există posibilitatea ca acelaşi factor să apară de mai multe ori (relaţia 2-28) ceea ce complică mult efectuarea operaţiei de descompunere cu ajutorul circuitelor logice.

In cazul utilizării logaritmilor binari într-un calculator electronic se prelucrează însă separat caracteristica şi mantisa acestora [1]. De aceea un număr oarecare N se poate aduce prin deplasări (în sisteme cu virgulă fixă) sau prin normalizare (în sisteme cu virgulă mobilă) în domeniul :

$$1 \leqslant A < 2 \tag{2-29}$$

legătura dintre A şi N fiind dată de relaţia :

$$N = 2^C A \tag{2-30}$$

Pentru sistemul binar numărul întreg C se poate uşor determina atît în cazul lucrului cu virgulă fixă (din cantitatea deplasărilor numărului N) cît şi în cazul lucrului cu virgulă mobilă (din exponentul numărului N).

Asupra caracteristicilor logaritmilor în cazul operaţiilor de înmulţire, împărţire, ridicare la putere şi extragerea rădăcinii se efectuează operaţii simple de adunare, scădere, deplasare [1],/71/. De aceea în cadrul tezei s-a studiat algoritmul de calcul al logaritmilor şi antilogaritmilor binari pentru un număr A cuprins în domeniul (2-29).

Pentru $1 \leqslant A < 2$ rezultă că în descompunerea (2-28) $A_0=1$. De asemenea din calcule rezultă că :

$$\log_2 A < 1 \tag{2-31}$$

$$\log_2(1+2^{-1}) > \frac{1}{2} \tag{2-32}$$

$$\log_2(1+2^{-i}) < 2\,\log_2\left[1+2^{-(i+1)}\right] \tag{2-33}$$

Pe baza acestor relaţii se poate trage concluzia că factorul ce conţine pe 2^{-1} din descompunerea (2-28) nu poate să apară decît o singură dată (altfel ar rezulta $\log_2 A > 1$) iar factorii următori nu pot nici ei să apară în descompunere decît o singură dată (altfel în locul a 2 factori de ordinul i+1 s-ar putea introduce un singur factor de ordinul i). Cu aceasta se poate pune dezvoltarea în factori a numărului binar A în forma :

$$A = (1+k_1 2^{-1})(1+k_2 2^{-2})\dots(1+k_n 2^{-n}) = \prod_{i=1}^{i=n}(1+k_i 2^{-i}) \tag{2-34}$$

BUPT

In relaţia (2-34) factorii pot avea valorile :

$$1 + k_i 2^{-i} \begin{cases} 1 \quad , \text{ cînd } k_i = 0 \\ 1+2^{-i}, \text{ cînd } k_i = 1 \end{cases}$$

In primul caz factorii nu apar în descompunere iar în al doilea caz ei nu au o valoare constantă cunoscută. Se poate vedea uşor că în descompunere nu pot să apară toţi factorii deoarece pentru $k_1 = k_2 = \ldots k_n = 1$ ar rezulta $A = 2,38$ ceea ce nu este posibil dacă numărul A este cuprins în domeniul (2-29).

Deci cel puţin un factor va lipsi din descompunere ($k_1 = 0$ sau $k_2 = 0$ cînd lipseşte un singur factor). Trebuie remarcat deasemenea că pentru $A \geq 1,5$ (în binar 1,1000...) întotdeauna $k_1 = 1$, deci $k_1 = a_1$, a_1 fiind bitul din dreapta virgulei numărului A, deoarece $1+1.2^{-1} = 1,1000\ldots$

Prin logaritmare în baza 2 a ultimei forme a numărului A rezultă :

$$L = \log_2 A = \log_2(1+k_1 2^{-1}) + \log_2(1+k_2 2^{-2}) + \ldots + \log_2(1+k_n 2^{-n}), \tag{2-35}$$

unde fiecare termen din dreapta poate avea valoarea

$$\log (1+k_i 2^{-i}) \begin{cases} 0 \quad , \text{ cînd } k_i = 0 \\ L_i = \log_2(1+2^{-i}), \text{ cînd } k_i = 1, \end{cases}$$

valorile L_i fiind constante cunoscute, calculate de autorul tezei în /73/ (Tabelul 2-2) cu 43 de biţi.

Relaţia (2-35) se poate scrie deci :

$$L = k_1 L_1 + k_2 L_2 + \ldots + k_n L_n = \sum_{i=1}^{i=n} k_i L_i \tag{2-36}$$

care dă logaritmul binar L al numărului A în funcţie de valorile coeficienţilor de creştere k_i.

Se poate enunţa acum algoritmul de calcul al logaritmului binar al unui număr A cuprins în domeniul (2-29). El include o operaţie de "descompunere în factori" şi apoi o operaţie de "însumare a termenilor" (prin termen se va înţelege logaritmul factorului).

1. Se compară numărul A cu factorul $(1+2^{-1})$ şi rezultă una din situaţiile :

   a. $A \geq (1+2^{-1})$, deci $k_1 = 1$
   b. $A < (1+2^{-1})$, deci $k_1 = 0$

2. Se compară numărul A cu produsul factorilor $(1+k_1 2^{-1})(1+2^{-2})$ şi rezultă una din situaţiile :

   a. $A \geq (1+k_1 2^{-1})(1+2^{-2})$, deci $k_2 = 1$
   b. $A < (1+k_1 2^{-1})(1+2^{-2})$, deci $k_2 = 0$

BUPT

n. Se compară numărul A cu produsul :

$$(1+k_1 2^{-1})\ldots\ldots\left[1+k_{n-1}2^{-(n-1)}\right](1+2^{-n})$$

şi rezultă una din situaţiile :

a. $A > (1+k_1 2^{-1})\ldots\left[1+k_{n-1}2^{-(n-1)}\right](1+2^{-n})$, deci $k_n = 1$

b. $A < (1+k_1 2^{-1})\ldots\left[1+k_{n-1}2^{-(n-1)}\right](1+2^{-n})$, deci $k_n = 0$.

Fiind determinate valorile coeficienţilor de creştere k_i se aplică relaţia (2-36) şi se determină logaritmul $L=\log_2 A$ căutat.

TABELUL 2-2 L_i în zecimal şi în binar

1	0,5849625007211	0,1001010111000000000110100011100111111011110
2	0,3219280948873	0,0101001001101001111000010010111100110100011
3	0,1699250014423	0,0010101110000000001101000111001111110111101
4	0,0874628412503	0,0001011001100011111101101111101011001001000
5	0,0443941193584	0,0000101101011101011010011011101011000111011
6	0,0223678130284	0,0000010110111001111001011010000101110000101
7	0,0112272554232	0,0000001011011111110010100001011011011101111
8	0,0056245491938	0,000000010111000010011100010001101101011101
9	0,0028150156070	0,000000001011100001111100000111111111000010
10	0,0014081943928	0,0000000001011100010010011001010011011101000
11	0,0007042690112	0,0000000000101110001001111010110001011110111
12	0,0003521774803	0,0000000000010111000101001000111011000010101
13	0,0001760994864	0,0000000000001011100010100111010110001000111
14	0,0000880524301	0,0000000000000101110001010100011001001110110
15	0,0000440268868	0,0000000000000010111000101010011000001010000
16	0,0000220136113	0,0000000000000001011100010101001110111101101
17	0,0000110068476	0,0000000000000000101110001010101000001100111
18	0,0000055034343	0,0000000000000000010111000101010100010010000
19	0,0000027517197	0,0000000000000000001011100010101010001011111
20	0,0000013758605	0,0000000000000000000101110001010101000110101
21	0,0000006879304	0,0000000000000000000010111000101010100011100
22	0,0000003439652	0,0000000000000000000001011100010101010001110
23	0,0000001719826	0,0000000000000000000000101110001010101000111
24	0,0000000859913	0,0000000000000000000000010111000101010100011
25	0,0000000429956	0,0000000000000000000000001011100010101010001
26	0,0000000214978	0,0000000000000000000000000101110001010101000
27	0,0000000107489	0,0000000000000000000000000010111000101010100
28	0,0000000053744	0,0000000000000000000000000001011100010101010
29	0,0000000026872	0,0000000000000000000000000000101110001010101
30	0,0000000013436	0,0000000000000000000000000000010111000101010
31	0,0000000006718	0,0000000000000000000000000000001011100010101
32	0,0000000003359	0,0000000000000000000000000000000101110001010
33	0,0000000001679	0,0000000000000000000000000000000010111000101
34	0,0000000000839	0,0000000000000000000000000000000001011100010
35	0,0000000000419	0,0000000000000000000000000000000000101110001
36	0,0000000000209	0,0000000000000000000000000000000000010111000
37	0,0000000000104	0,0000000000000000000000000000000000001011100
38	0,0000000000052	0,0000000000000000000000000000000000000101110
39	0,0000000000026	0,0000000000000000000000000000000000000010111
40	0,0000000000013	0,0000000000000000000000000000000000000001011
41	0,0000000000006	0,0000000000000000000000000000000000000000101
42	0,0000000000003	0,0000000000000000000000000000000000000000010
43	0,0000000000001	0,0000000000000000000000000000000000000000001

INSTITUTUL POLITEHNIC
TIMIŞOARA
CENTRALĂ

BUPT

Algoritmul de calcul al antilogaritmului binar al unui logaritm L dat, cuprins în domeniul :

$$0 \leq L < 1 \qquad (2\text{-}37)$$

constă din parcurgerea în sens invers a algoritmului de calcul al logaritmului binar de mai sus. Algoritmul include operația de "descompunere în termeni" a logaritmului și apoi operația de "înmulțire a factorilor" (factorii reprezintă antilogaritmii termenilor) :

1. Se compară logaritmul L cu constanta L_1 și rezultă una din situațiile :

   a. $L \geq L_1$, deci $k_1 = 1$
   b. $L < L_1$, deci $k_1 = 0$

2. Se compară logaritmul L cu suma $kL_1 + L_2$ și rezultă una din situațiile :

   a. $L \geq k_1 L_1 + L_2$, deci $k_2 = 1$
   b. $L < k_1 L_1 + L_2$, deci $k_2 = 0$

.
.
.

n. Se compară logaritmul L cu suma :

$$k_1 L_1 + \ldots + k_{n-1} L_{n-1} + L_n$$

și rezultă una din situațiile :

   a. $L \geq k_1 L_1 + \ldots + k_{n-1} L_{n-1} + L_n$, deci $k_n = 1$
   b. $L < k_1 L_1 + \ldots + k_{n-1} L_{n-1} + L_n$, deci $k_n = 0$

Fiind determinate valorile coeficienților de creștere (operația de descompunere în termeni) se calculează acum antilogaritmul A cu relația (2-34) ceea ce constituie operația de înmulțire a factorilor.

Se observă și aici că nu toți coeficienții de creștere pot fi egali cu 1 deoarece suma tuturor constantelor L_i conduce la un număr mai mare decît 1 (cca.1,25).

Operațiile de descompunere în factori și însumare a termenilor la calculul logaritmului se pot desfășura și în paralel, pe măsură ce se determină coeficienții de creștere. In mod asemănător, descompunerea în termeni și înmulțirea factorilor se pot desfășura în paralel.

Operațiile aritmetice cerute de algoritmul prezentat constă deci din comparări (efectuate de exemplu prin determinarea împrumutului din poziția cea mai semnificativă la scădere), adunări și înmulțiri. Inmulțirea se face însă cu un factor de forma $(1+2^{-i})$ și se reduce deci la o adunare a produsului parțial cu el însuși deplasat însă cu i poziții binare spre dreapta.

- Se poate observa că algoritmul de logaritmare și cel de

BUPT

antilogaritmare includ acelaşi tip de operaţii elementare ceea ce permite utilizarea, cu mici modificări a aceluiaşi hardware la generarea logaritmilor şi antilogaritmilor binari.

Calculul logaritmilor şi antilogaritmilor binari conform algoritmului de mai sus reprezintă un calcul iterativ în care numărul de iteraţii este fixat dinainte prin adoptarea cantităţii n de cifre ale factorilor $(1+2^{-i})$ şi constantelor L_i. In aceste iteraţii apar o serie de erori şi anume :

- la calculul logaritmului :

1. eroarea absolută datorată descompunerii în factori a cărei limită superioară şi valoare posibilă se notează cu ΔL_{1l} respectiv ΔL_{1p},

2. eroarea absolută datorată cantităţii finite de cifre a termenilor L_i, a cărei limită superioară şi valoare posibilă se notează cu ΔL_{2l} respectiv ΔL_{2p} /58/.

- la calculul antilogaritmului :

3. eroarea absolută datorată descompunerii în termeni, a cărei limită superioară şi valoare posibilă se notează cu ΔA_{1l} respectiv ΔA_{1p},

4. eroarea absolută datorată rotunjirilor la înmulţirea factorilor, a cărei limită superioară şi valoare posibilă se notează cu ΔA_{2l} respectiv ΔA_{2p} /58/.

1. Eroarea ce apare la descompunerea în factori se datoreşte faptului că din cauza numărului de biţi limitat, pentru un ordin $i \geq \frac{n}{2}+1$ (uneori chiar pentru un ordin mai mic cînd termenul L_{i-1} are la sfîrşit un număr mare de zerouri consecutive) dispare inegalitatea (2-33). Cu alte cuvinte, apare situaţia :

$$2\,L_i \leqslant L_{i-1} \qquad (2\text{-}38)$$

Aceasta înseamnă că un factor de ordin $i \geq \frac{n}{2}+1$ ar putea să fie utilizat de două ori în descompunerea în factori fără ca acest lucru să ducă la transformarea factorului dedublat într-un factor simplu de ordin i-1 cum s-a arătat anterior. Daca nu există posibilitatea de a se sesiza apariţia unui factor dedublat în descompunere atunci un factor $(1+2^{-i})$ este "scăpat". In continuare compensarea totală a absenţei lui prin apariţia unor factori de ordin mai mare decît i nu mai este posibilă din cauză că :

$$(1+2^{-i}) > \prod_{i=i+1}^{i=n} (1+2^{-i}) \qquad (2\text{-}39)$$

Aceasta face ca la sfîrşitul descompunerii să nu existe în totdeauna identitatea :

$$\prod_{i=1}^{i=n} (1+k_i 2^{-i}) = A \qquad (2\text{-}4o)$$

INSTITUTUL POLITEHNIC TIMIŞOARA BIBLIOTECA CENTRALĂ

BUPT

Eroarea la descompunerea în factori este maximă cînd este "scăpat" un factor de ordinul $i = \frac{n}{2} + 1$ (de unde începe să se manifeste inegalitatea (2-38). Astfel, în locul operației :

$$P_{\frac{n}{2}+1} \cdot \left[1 + 2^{-(\frac{n}{2}+1)}\right] \qquad (2\text{-}41)$$

unde $P_{\frac{n}{2}+1}$ reprezintă produsul $\prod_{i=1}^{\frac{n}{2}+1} (1+2^{-i})$, se efectuează la descompunere operația :

$$P_{\frac{n}{2}+1} \cdot \prod_{i=\frac{n}{2}+2}^{i=n} (1+2^{-i}) \qquad (2\text{-}42)$$

Comparînd acum relațiile (2-41) și (2-42) se observă că deoarece între primul produs și al doilea există o diferență, mai este posibilă, înafară de scăparea factorului de ordin $\frac{n}{2}+1$, scăparea a încă unui factor al cărui ordin este dat de diferența :

$$\left[1+2^{-(\frac{n}{2}+1)}\right] - \prod_{i=\frac{n}{2}+2}^{i=n} (1+2^{-i}) <$$

$$< \left[1+2^{-(\frac{n}{2}+1)}\right] - (1 + \sum_{i=\frac{n}{2}+2}^{i=n} 2^{-i}) = 2^{-n} \qquad (2\text{-}43)$$

Prin urmare mai poate fi scăpat la descompunere un factor de ordinul n adică $(1+2^{-n})$.

Scăparea celor doi factori, arătați mai sus, conduce în cadrul operației de însumare a logaritmilor factorilor la o eroare absolută limită :

$$\Delta L_{11} = \log_2 \left[1+2^{-(\frac{n}{2}+1)}\right] + \log_2(1-2^{-n}) - \log \prod_{i=\frac{n}{2}+2}^{i=n} (1+2^{-i}) =$$

$$= L_{\frac{n}{2}+1} - \sum_{i=\frac{n}{2}+2}^{i=n} L_i + L_n = n_1 . 2^{-n} + 2^{-n} \qquad (2\text{-}44)$$

Efectuînd aceste calcule pe baza Tabelului 2-3 se observă că rezultatul depinde de cantitatea n_1 de biți egali cu 1 a constantelor L_i (pentru $i \geq \frac{n}{2}+1$) în poziția 2^{-n}. Dacă se consideră cazul teoretic cînd toate constantele L_i au în ultima poziție un bit 1 rezultă eroarea limită :

$$\Delta L_{11} < (\frac{n}{2}+1)\, 2^{-n} + 2^{-n} = n . 2^{-(n+1)} \qquad (2\text{-}45)$$

Eroarea absolută posibilă este însă de două ori mai mică deoarece, în general, numai jumătate din constantele L_i au ultimul bit egal cu 1 și deasemenea este "scăpat" un singur factor :

$$L_{1p} \cong \frac{n}{4} . 2^{-n} = n . 2^{-(n+2)} \qquad (2.46)$$

BUPT

Pentru $n = 28$, relaţia (2-46) conduce la o eroare :

$$\Delta L_{1p} = 2^{-26} + 2^{-27} + 2^{-28} = 2^{-25} \qquad (2\text{-}47)$$

Aplicarea pentru acest caz a relaţiei (2-44) ţinînd cont de Tabelul 2-2 conduce la rezultatul :

$$\Delta L_{1p} = \Delta L_{11} = 2^{-25} \qquad (2\text{-}48)$$

In concluzie eroarea absolută cauzată de descompunerea în factori afectează numai ultimele 3-4 poziţii ale logaritmului. Ea depinde de faptul dacă la limitarea cantităţii de cifre ale termenilor L_i rămîne în ultima poziţie o cantitate mai mare sau mai mică de biţi 1 (Tabelul 2-2).

2. Eroarea absolută datorată cantităţii finite de biţi ai termenilor L_i se poate scrie (relaţia 2-36) :

$$L_2 = \sum_{i=1}^{i=n} k_i e_r \qquad (2\text{-}49)$$

unde e_r este eroarea de rotunjire a termenilor L_i la n biţi :

$$e_r < 2^{-n} \qquad (2\text{-}50)$$

Deoarece cel puţin un coeficient k_i este nul /58/ rezultă eroarea absolută limită :

$$\Delta L_{21} < (n-1)2^{-n} \qquad (2\text{-}51)$$

In realitate nu toţi termenii L_i au aceeaşi eroare de rotunjire e_r. Din Tabelul 2-2 se poate vedea că, prin reţinerea unei cantităţi de biţi n ai termenilor, aproximativ jumătate din termeni au eroarea de rotunjire $e_r < 2^{-n}$ (se înlătură biţi 1 din poziţia $2^{-(n+1)}$ iar cealaltă jumătate are eroarea de rotunjire $e_r < 2^{-(n+1)}$ (se înlătură biţi 1 din poziţia $2^{-(n+2)}$). Cu aceasta rezultă eroarea absolută posibilă (un coeficient k_i se consideră nul) :

$$\Delta L_{2p} = \frac{n}{2} \cdot 2^{-n} + \left(\frac{n}{2} - 1\right) 2^{-(n+1)} = (3n-2) \cdot 2^{-(n+2)} \qquad (2\text{-}52)$$

In majoritatea cazurilor eroarea ΔL_{2p} va fi mai mică decît cea dată de relaţia de mai sus deoarece erorile de rotunjire sînt mai mici decît cele admise iar coeficienţii k_i nu sînt toţi egali cu 1.

Eroarea absolută limită în cadrul operaţiei de calcul al algoritmilor va fi deci :

$$\Delta L_1 = \Delta L_{11} + \Delta L_{21} \quad n \cdot 2^{-(n+1)} + (n-1) \cdot 2^{-n} = (3n-2) \cdot 2^{-(n+1)} \qquad (2\text{-}53)$$

iar eroarea absolută posibilă :

$$\Delta L_p = \Delta L_{1p} + \Delta L_{2p} = n \cdot 2^{-(n+2)} + (3n-2) \cdot 2^{-(n+2)} = (2n-1) 2^{-(n+1)} \qquad (2\text{-}54)$$

BUPT

Pentru un număr de biţi n=28 rezultă astfel :

$$\Delta L_1 = 2^{-23} + 2^{-25} + 2^{-27} \qquad (2\text{-}55)$$

$$\Delta L_p = 2^{-24} + 2^{-25} + 2^{-26} \qquad (2\text{-}56)$$

Deoarece cazul n=28 s-a analizat în mod special în teză s-a calculat cu mare precizie eroarea absolută maximă pe baza tabelului constantelor L_i cu 43 de biţi, în cazul cînd n-1 coeficienţi k_i sînt egali cu 1, rezultînd :

$$\Delta L_l = \Delta L_p = 2^{-24} + 2^{-26} + 2^{-27} \qquad (2\text{-}57)$$

care este apropiată de cea dată de relaţia teoretică (2-56). Deci eroarea absolută maximă afectează ultimele cinci poziţii ale logaritmului pentru n=28.

3. La calculul antilogaritmului eroarea absolută datorată descompunerii în termeni survine din aceleaşi motive ca şi cea de la descompunerea în factori. Cu alte cuvinte, în locul apariţiei de două ori în descompunere a unui termen $L_{\frac{n}{2}+1}$ descompunerea va conţine un singur termen $L_{\frac{n}{2}+1}$ şi suma de termeni

$$\sum_{i=\frac{n}{2}+2}^{i=n} L_i < L_{\frac{n}{2}+1} \qquad (2\text{-}58)$$

care încearcă să compenseze "scăparea termenului $L_{\frac{n}{2}+1}$. Aceasta face ca la sfîrşitul descompunerii să nu existe în totdeauna identitatea :

$$\sum_{i=1}^{i=n} k_i L_i = L \qquad (2\text{-}59)$$

(compensarea nu reuşeşte). Existenţa unei diferenţe între membrii inegalităţii (2-58) face posibilă scăparea a încă unei serii de termeni a căror sumă este egală cu diferenţa amintită. Factorii corespunzători termenilor scăpaţi nu vor apărea în operaţia de înmulţire şi conduc la eroare în calculul antilogaritmului.

Diferenţa se poate calcula cu relaţia :

$$L_{\frac{n}{2}+1} - \sum_{i=\frac{n}{2}+2}^{i=n} L_i = n_1 . 2^{-n} \qquad (2\text{-}60)$$

unde n_1 reprezintă numărul de constante de ordin $i \geq \frac{n}{2}+1$ care au ultimul bit egal cu 1. Dacă se consideră cazul limită cînd toate constantele L_i au ultimul bit egal cu 1 rezultă o diferenţă :

$$L_{\frac{n}{2}+1} - \sum_{i=\frac{n}{2}+2}^{i=n} L_i = (\tfrac{n}{2} - 1).2^{-n} = (n-2).2^{-(n+1)} \qquad (2\text{-}61)$$

BUPT

căreia îi corespunde o sumă de termeni :

$$(n-2)2^{-(n+1)} \leqslant L_{n+1-m} + L_{n+1-(m-2)} \qquad (2\text{-}62)$$

unde m este numărul maxim ce îndeplineşte condiţia :

$$2^{m} \leqslant n \qquad (2\text{-}63)$$

Cu aceasta rezultă eroarea absolută limită la calculul antilogaritmului datorată descompunerii în termeni :

$$\Delta A_{11} \leqslant P_{\frac{n}{2}+1} \cdot \left[1+2^{-(\frac{n}{2}+1)}\right]\left[1+2^{-(n+1-m)}\right]\left[1+2^{-[(n+1)-(m-2)]}\right] -$$

$$- P_{\frac{n}{2}+1} \cdot \prod_{i=\frac{n}{2}+2}^{i=n} (1+2^{-i}) \qquad (2\text{-}64)$$

care, admiţînd că $P_{\frac{n}{2}+1} < 2$, obţine valoarea :

$$\Delta A_{11} < 2\left[1+2^{-(\frac{n}{2}+1)}+2^{-(n+1-m)}+2^{-(n+3-m)}-1 - \sum_{i=\frac{n}{2}+2}^{i=n} 2^{-i}\right] =$$

$$= 2^{-(n-m)} + 2^{-(n+2-m)} + 2^{-(n-1)} \qquad (2\text{-}65)$$

Eroarea absolută posibilă rezultă în cazul în care se consideră că numai jumătate din constantele L_i pentru $i \geqslant \frac{n}{2}+1$ au ultimul bit egal cu 1. Diferenţa (2-60) rezultă în acest caz :

$$L_{\frac{n}{2}+1} - \sum_{i=\frac{n}{2}+2}^{i=n} L_i = \frac{n}{4} \cdot 2^{-n} = n.2^{-(n+2)} \qquad (2\text{-}66)$$

căreia îi corespunde o sumă de termeni "scăpaţi" în plus :

$$n.2^{-(n+2)} \leqslant L_{n+2-m} + L_{n+2-(m-1)} = L_{n+2-m} + L_{n+3-m} \qquad (2\text{-}67)$$

Cu aceasta eroarea absolută posibilă devine :

$$\Delta A_{1p} = P_{\frac{n}{2}+1} \left[1+2^{-(\frac{n}{2}+1)}\right]\left[1+2^{-(n+2-m)}\right]\left[1+2^{-(n+3-m)}\right] -$$

$$- P_{\frac{n}{2}+1} \cdot \prod_{i=\frac{n}{2}+2}^{i=n} (1+2^{-i}) <$$

$$< 2\left[1+2^{-(\frac{n}{2}+1)}+2^{-(n+2-m)}+2^{-(n+3-m)}-1 - \sum_{i=\frac{n}{2}+2}^{i=n} 2^{-i}\right] =$$

$$= 2^{-(n+1-m)} + 2^{-(n+2-m)} + 2^{-(n-1)} \qquad (2\text{-}68)$$

4. Eroarea absolută limită datorată rotunjirilor la înmulţirea factorilor (relaţia 2-34) apare în cazul cînd toţi coeficienţii k_1 sînt egali cu 1 (de fapt cel puţin un coeficient dintre k_1

BUPT

şi k_2 este nul). Erorile prin rotunjire apar din cauza înlăturării biţilor din dreapta poziţiei 2^{-n} prin deplasarea produsului anterior /58/. Această înlăturare începe la înmulţirea de ordinul $i = s+1$ unde s este dat de relaţia :

$$2^{-(1+2+3+\ldots+s)} \geq 2^{-n}$$

sau :

$$1+2+3+\ldots+s \leqslant n \qquad (2\text{-}69)$$

Admiţînd că eroarea de rotunjire ce apare este e_r 2^{-n} rezultă eroarea absolută limită :

$$\Delta A_{21} = (n-s)e_r < (n-s)2^{-n} \qquad (2\text{-}70)$$

Ţinînd cont că în realitate eroarea de rotunjire este $e_r < 2^{-n}$ la jumătate din produsele deplasate şi $e_r < 2^{-(n+1)}$ la cealaltă jumătate, se obţine eroarea absolută posibilă :

$$\Delta A_{2p} = \frac{n-s}{2}\, 2^{-n} + \frac{n-s}{2}\, 2^{-(n+1)} = 3(n-s).2^{-(n+2)} \qquad (2\text{-}71)$$

Insumînd cele două categorii de erori ce apar la operaţia de calcul a antilogaritmului rezultă :

- eroarea absolută limită :

$$\Delta A_1 = \Delta A_{11} + \Delta A_{21} < 2^{-(n-m)} + 2^{-(n+2-m)} + 2^{-(n-1)} + (n-s)^{-n} =$$
$$= (n+2-s+2^m+2^{m-2}).2^{-n} \qquad (2\text{-}72)$$

şi eroarea absolută posibilă :

$$\Delta A_p = \Delta A_{1p} + \Delta A_{2p} = 2^{-(n+1-m)} + 2^{-(n+2-m)} + 2^{-(n-1)} + 3(n-s)2^{-(n+2)}$$
$$= (2^{m+1}) + 2^m + 3n+8-3s)\, 2^{-(n+2)} \qquad (2\text{-}73)$$

Pentru un număr de biţi $n=28$ rezultă $m=4$, $s=7$ şi se obţin din relaţiile de mai sus :

$$\Delta A_1 < 2^{-23} + 2^{-25} + 2^{-27} + 2^{-28} \qquad (2\text{-}74)$$

$$\Delta A_p = 2^{-24} + 2^{-25} + 2^{-26} + 2^{-28} \qquad (2\text{-}75)$$

Totuşi pentru cazul $n=28$ s-a determinat cu precizie mare eroarea absolută maximă :

$$\Delta A_1 = \Delta A_p = 2^{-24} + 2^{-26} + 2^{-27} + 2^{-28} \qquad (2\text{-}76)$$

care este aproximativ aceeaşi cu cea dată de relaţia (2-73) şi cu cea de la calculul logaritmului (relaţia 2-57).

Din relaţiile (2-57) şi (2-76) se vede că eroarea afectează numai ultimele poziţii binare ale logaritmului sau antilogaritmului (5 poziţii pentru n=28 biţi în cazul cel mai defavorabil).

Pentru asigurarea unei precizii mai bune într-un sistem ce generează logaritmi şi antilogaritmi trebuie folosite circuite

BUPT

pentru o cantitate de biţi n mai mare decît cantitatea de biţi a numerelor ce se logaritmează pentru ca eroarea de calcul să se poată neglija.

Astfel, pentru o unitate aritmetică cu virgulă mobilă logaritmică ce operează cu numere disponibile în calculator cu 24 biţi (un bit în faţa virgulei şi 23 după virgulă), pentru ca eroarea să fie mai mică decît 2^{-23} sînt necesare circuite de generare a logaritmului şi antilogaritmului pentru 28 biţi.

In urma studiului erorilor prezentat mai sus rezultă o serie de posibilităţi de reducere a acestora. Deoarece erorile absolute sînt întotdeauna negative o posibilitate de reducere la jumătate a erorii absolute posibile constă în adăugarea la rezultat a unei corecţii :

$$\frac{\Delta L_p}{2} \text{ la calculul logaritmulor}$$

$$\frac{\Delta A_p}{2} \text{ la calculul antilogaritmilor}$$

Deoarece principala eroare absolută apare din cauza cantităţii finite de biţi ai termenilor L_i şi ai produselor parţiale deplasate P_i există posibilitatea de a se reduce erorile prin aplicarea unei rotunjiri a acestora. Valorile L_i se pot rotunji în tabel iar valorile P_i prin schema electronică.

Pentru reducerea erorilor este indicată alegerea cantităţii n de biţi ţinînd cont şi de faptul ca numărul de termeni L_i ce conţin biţi 1 în poziţiile 2^{-n} şi $2^{-(n+1)}$ să fie cît mai redus.

In fine, eliminarea erorilor datorate descompunerii în factori sau în termeni este posibilă prin repetarea ultimului factor $(1+2^{-n})$ sau a ultimului termen L_n pînă cînd se obţine identitatea :

$$\prod_{i=1}^{i=n} (1+k_i 2^{-i}) = A$$

respectiv : (2-77)

$$\sum_{i=1}^{i=n} k_i L_i = L$$

Pentru reducerea numărului de factori la descompunere s-a analizat deasemenea posibilitatea utilizării unor factori de forma

$$1 + k_i\, 2^{-i} + 2^{-(i+1)} \tag{2-78}$$

Acest lucru necesită însă însumarea simultană a trei numere (produsul anterior, produsul deplasat cu i respectiv i+1 poziţii binare) pentru care sînt necesare sumatoare cu 4 intrări. Deasemenea, prin folosirea unor factori mai mari va rezulta o eroare suplimen-

BUPT

tară de cuantificare.

Algoritmul de calcul al logaritmilor şi antilogaritmilor binari bazat pe descompunerea în factori, propus de autorul tezei, are următoarele avantaje :

- simplitatea operaţiilor aritmetice necesare în calcule,
- utilizarea aceloraşi operaţii aritmetice atît la calculul logaritmului cît şi al antilogaritmului: adunări, comparări şi deplasări,
- asigurarea preciziei dorite prin simpla extindere a cantităţii de biţi ai numerelor,
- reducerea la minimum a numărului de iteraţii necesar, faţă de alţi algoritmi,
- uşurinţa transpunerii algoritmului într-un hardware pentru generarea logaritmilor şi antilogaritmilor binari, care poate fi o unitate aritmetică obişnuită sau una ce include structuri logice celulare de logaritmare şi antilogaritmare,
- posibilitatea realizării pe baza algoritmului a unei subrutine simple pentru calculatoare numerice,
- aplicabilitatea simplă în cazul sistemului cu virgulă mobilă.

Ca dezavantaj al algoritmului propus se poate considera acela al necesităţii efectuării calculelor cu o cantitate de biţi mai mare decît aceea a numărului ce se logaritmează cînd este impusă o eroare mai mică decît bitul cel mai puţin semnificativ al numărului. Aceasta conduce la creşterea cantităţii circuitelor necesare în schema de generare.

Intrucît între algoritmul propus de autorul tezei /37/,/58/ şi algoritmul Perle [24] există unele asemănări şi au fost comunicaţi în acelaşi an în mod independent, se prezintă mai jos deosebirile dintre ei.

- algoritmul propus de autorul tezei pleacă de la un algoritm cunoscut pentru calculul cu tabele al logaritmului zecimal [3],
- algoritmul propus de autorul tezei este concretizat pentru logaritmi binari unde determinarea caracteristicii este simplă, în timp ce algoritmul Perle este generalizat pentru o bază b oarecare ,
- algoritmul propus de autorul tezei operează numai cu partea fracţionară a numărului şi cu mantisa logaritmului în timp ce algoritmul Perle include la calcule şi partea întreagă a numărului respectiv caracteristica logaritmului ceea ce complică mult calculele şi circuitele,

BUPT

- algoritmul prbpus de autorul tezei studiază posibilitatea apariției multiple a unui factor (sau termen) în iterație și se concretizează la baza 2 cînd în iterație factorii sînt uniei în timp ce algoritmul Perle pentru a putea fi utilizat la o bază $b>2$ necesită repetarea de un număr de ori a unei iterații,
- algoritmul Perle [24] a avut o greşeală pe care autorul tezei a sesizat-o în /72/ privind convergența iterației pentru o bază $b>2$; deasemenea autorul tezei a propus completarea algoritmului Perle în acest sens /72/,
- algoritmul propus de autorul tezei include numai operații de deplasare, adunare și comparare în timp ce algoritmul Perle include în plus operații de scădere la calculul antilogaritmului,
- algoritmul propus de autorul tezei este însoțit de un studiu al erorilor pentru cazul cînd cantitatea iterațiilor este impusă de cantitatea de biți ai numerelor și a registrelor din echipamentul de calcul /58/,
- algoritmul propus de autorul tezei conține valorile termenilor L_i calculate cu 43 cifre binare /73/,/37/,
- autorul tezei a făcut și propuneri de structuri logice celulare pentru generarea logaritmilor și antologaritmilor binari /37/.

2.3. Calculul constantelor L_i și C_i cu ajutorul calculatorului electronic.

Calculul șirurilor de constante L_i și C_i /73/ s-a efectuat cu metoda prezentată în [3], folosindu-se tabelul de logaritmi zecimali cu 32 cifre dat acolo. Pentru calcul s-a utilizat calculatorul numeric IRIS-50, programele fiind scrise în FORTRAN /73/. In calculator, în cazul lucrului cu precizie dublă, lungimea părții fracționare a numerelor a fost de 17 cifre zecimale. Eroarea cu care au fost cunoscute numerele în calculator a fost mai mică de 1.10^{-16} (cifra a 17-a nu a mai fost corectă).

Tabelul de logaritmi folosit conține logaritmii în baza 10 ai unor factori zecimali de forma dată de relația (2-79) iar metoda de calcul se bazează pe descompunerea în astfel de factori și adunarea logaritmilor factorilor (adunarea termenilor) în cazul calculului logaritmilor L_i sau pe descompunerea în termeni și înmulțirea antilogaritmilor termenilor (înmulțirea factorilor) în cazul calculului constantelor C_i.

Factorii zecimali utilizați în calcule au forma :

$$(1 + k_x 10^{-n}) \qquad (2\text{-}79)$$

BUPT

în care $n=2...16$, $x=1...3$, $k_1=5$, $k_2=2$, $k_3=1$ iar în tabelul din [3] sînt date valorile :

$$\log_{10}(1 + k_x 10^{-n}) \qquad (2\text{-}80)$$

Pentru calculul unei constante $L_i = \log_2(1+2^{-i})$ se compară $(1+2^{-i})$ cu $(1+5.10^{-2})$ - primul factor din tabelul dat [3]. Dacă :

$$(1+2^{-i}) \geq (1+5.10^{-2}) \qquad (2\text{-}81)$$

atunci factorul $(1+5.10^{-2})$ face parte din descompunerea în factori a lui $(1+2^{-i})$ şi logaritmul său :

$$\log_{10}(1+5.10^{-2}) \qquad (2\text{-}82)$$

este adunat într-un registru al rezultatului L_i. Se verifică în continuare dacă acelaşi factor $(1+5.10^{-2})$ mai apare încă odată în descompunerea în factori a lui $(1+2^{-i})$ făcîndu-se comparaţia între

$$(1+2^{-i}) \quad \text{şi} \quad (1+5.10^{-2})\,(1+5.10^{-2})$$

iar dacă acest produs nu depăşeşte numărul $(1+2^{-i})$ se adună din nou logaritmul factorului $(1+5.10^{-2})$ la rezultat. Operaţiile se repetă pînă ce factorul $(1+5.10^{-2})$ nu mai apare în descompunere. Se trece atunci la factorul următor - $(1+2.10^{-2})$ - mărindu-se x cu 1, cu care se efectuează aceleaşi operaţii de înmulţire şi comparare, adunîndu-se dacă este cazul logaritmul factorului la rezultat. Calculul continuă prin creşterea lui x şi n pînă la verificarea prezenţei în descompunere a factorului $(1+1.10^{-16})$. Suma logaritmilor factorilor incluşi în descompunere va reprezenta cu aproximaţie pe :

$$\log_{10}(1+2^{-i}) \qquad (2\text{-}83)$$

care se converteşte simplu în $L_i = \log_2(1+2^{-i})$.

Eroarea absolută de calcul se datoreşte în special însumării erorilor logaritmilor factorilor (2-80) care sînt cunoscuţi cu o eroare mai mică de 1.10^{-16}. In cadrul programului de calcul s-a prevăzut şi numărarea cantităţii j de adunări efectuate pentru obţinerea logaritmului L_i. Eroarea absolută la calcul va fi de ordinul $j.10^{-16}$ şi va afecta teoretic 2-3 cifre ale constantelor L_i astfel că numai 13-14 cifre zecimale ale constantelor sînt exacte (43-45 cifre binare).

In scopul calculării constantelor C_i acestea se scriu :

$$C_i = \text{antilog}_{10}(2^{-i}\log_{10}2) \qquad (2\text{-}84)$$

şi pentru efectuarea calculului logaritmul $(2^{-i}\log_{10}2)$ se descompune în termeni de forma (2-80) daţi în [3], urmînd să se înmulţească apoi antilogaritmii acestor termeni - adică factorii de forma (2-79) folosiţi şi anterior. Calculul decurge la fel ca în

BUPT

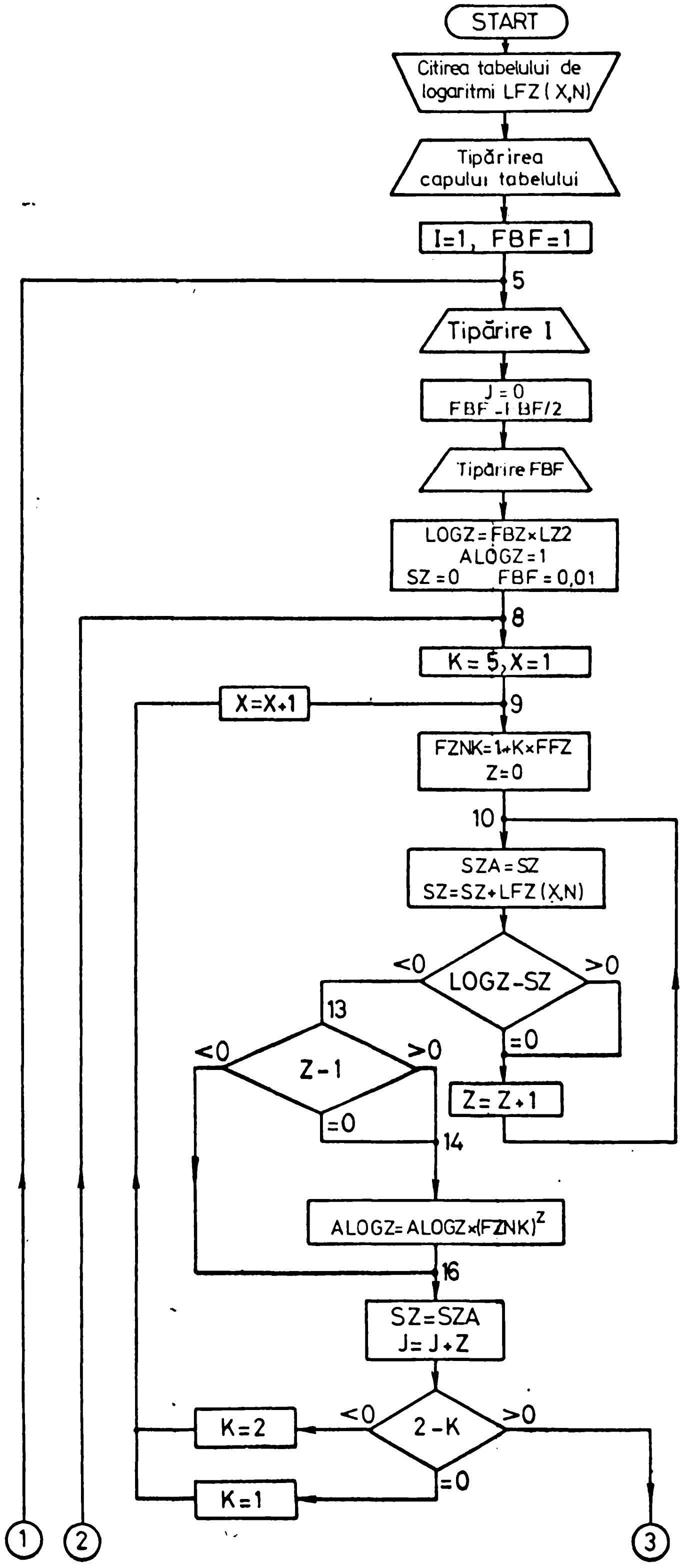
cazul constantelor L_i iar erorile absolute sînt de acelaşi ordin de mărime /73/.

Ca o verificare a preciziei de calcul s-a comparat constanta $C_1 = 2^{2^{-1}} = \sqrt{2}$, cunoscută cu mai mult de 17 cifre secimale exacte [3], cu valoarea ei calculată cu calculatorul electronic. Rezultă că C_1 s-a determinat cu 15 cifre zecimale exacte deci eroarea absolută a rezultat cu un ordin de mărime mai mică decît cea teoretică presupusă mai sus.

Constantele C_i şi L_i sînt prezentate cu 13 cifre zecimale exacte după virgulă în Tabelul 2-1 şi Tabelul 2-2, pentru i=1...43. Limita superioară a indicelui i reprezintă cu aproximaţie numărul de cifre binare corespunzătoare celor 13 cifre zecimale ale constantelor. Din tabele se observă că de la mijloc în jos o constantă se obţine din cea anterioară prin deplasare spre dreapta cu o poziţie.

In fig.2-1 şi 2-2 se prezintă ordinogramele programelor de calcul a constantelor C_i şi L_i. In acestea s-au folosit următoarele notaţii principale :

LFZ (K,N) = $\log_{10}(1+k_x 10^{-n})$ = logaritmul factorului zecimal,
FB = $(1+2^{-i})$ = factorul binar,
FBF = 2^{-i} = partea fracţionară a factorului binar,
FFZ = 10^{-n} = partea fracţionară a factorului zecimal,
FZNK = $(1+k.10^{-n})$ = factorul zecimal,
SZ = suma de logaritmi zecimali,
SZA = suma de logaritmi zecimali anterioară,
LZ2 = $\log_{10}2$,
LOGZ = $2^{-i}\log_{10}2$ = logaritmul zecimal al lui Ci = $2^{2^{-i}}$,
ALOGZ = antilog_{10}(LOGZ), care în final devine C_i,
CB = cifra binară a lui C_i,
FZA = produsul de factori zecimali anterior ,
FZ = produsul factorilor zecimali,
LOGZ = suma logaritmilor zecimali ai factorilor care în final
devine L_i în baza 10,
LOGBZ = L_i în baza 2 în zecimal.


START
Citirea tabelului de logaritmi LFZ (X,N)
Tipărirea capului tabelului
I=1, FBF=1
5
Tipărire I
J = 0
FBF -I BF/2
Tipărire FBF
LOGZ=FBZ×LZ2
ALOGZ=1
SZ=0 FBF=0,01
8
K=5, X=1
X=X+1
9
FZNK=1+K×FFZ
Z=0
10
SZA=SZ
SZ=SZ+LFZ(X,N)
LOGZ-SZ
<0
>0
=0
Z=Z+1
13
Z-1
<0
>0
=0
14
ALOGZ=ALOGZ×(FZNK)^Z
16
SZ=SZA
J=J+Z
2-K
<0
>0
=0
K=2
K=1
1
2
3

BUPT

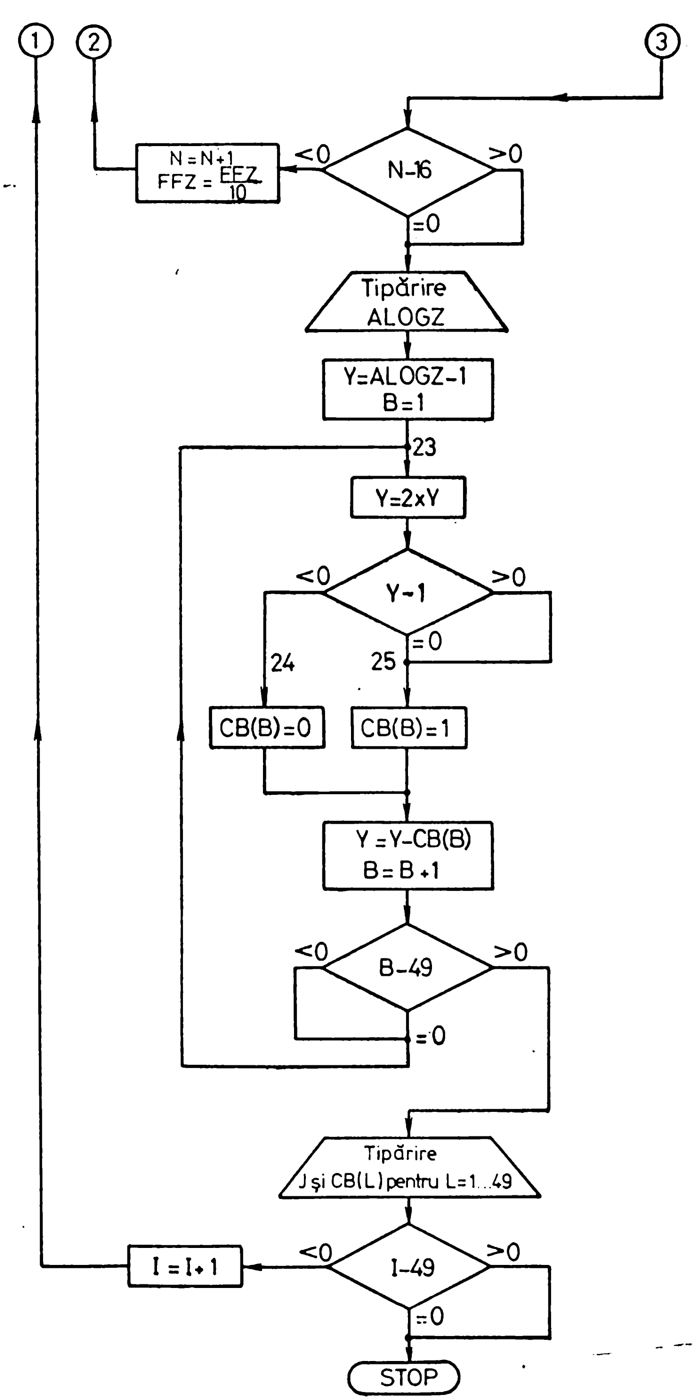
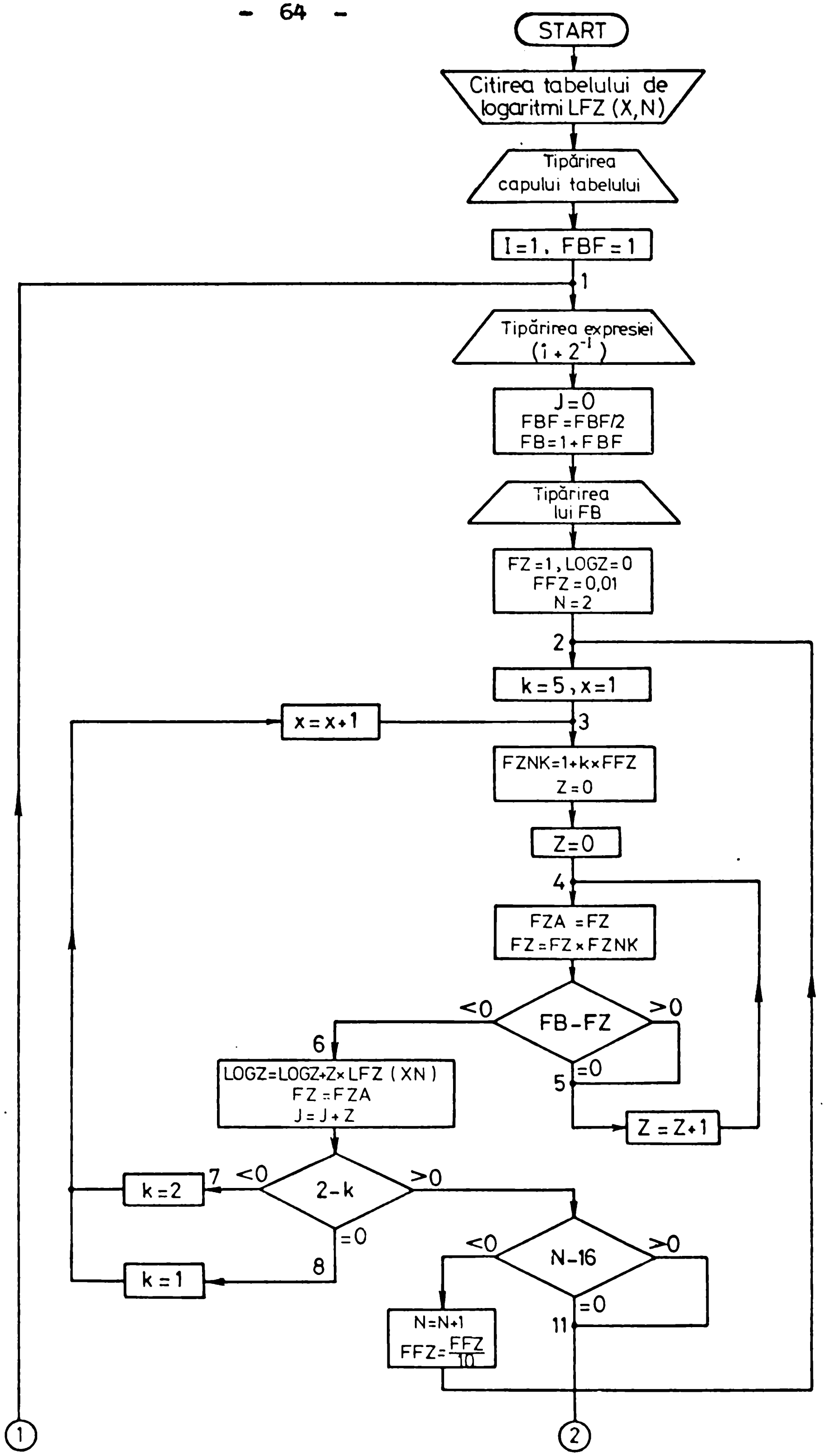


Fig.2-1 Ordinograma programului pentru calculul constantelor C_i.

BUPT


START
Citirea tabelului de logaritmi LFZ (X,N)
Tipărirea capului tabelului
I=1, FBF=1
1
Tipărirea expresiei (i + 2^-i)
J=0
FBF=FBF/2
FB=1+FBF
Tipărirea lui FB
FZ=1, LOGZ=0
FFZ=0,01
N=2
2
k=5, x=1
x=x+1
3
FZNK=1+k×FFZ
Z=0
Z=0
4
FZA =FZ
FZ=FZ×FZNK
FB-FZ
<0
>0
=0
5
6
Z=Z+1
LOGZ=LOGZ+Z×LFZ (XN)
FZ=FZA
J=J+Z
2-k
<0
>0
=0
7
8
k=2
k=1
N-16
<0
>0
=0
11
N=N+1
FFZ=FFZ/10
1
2

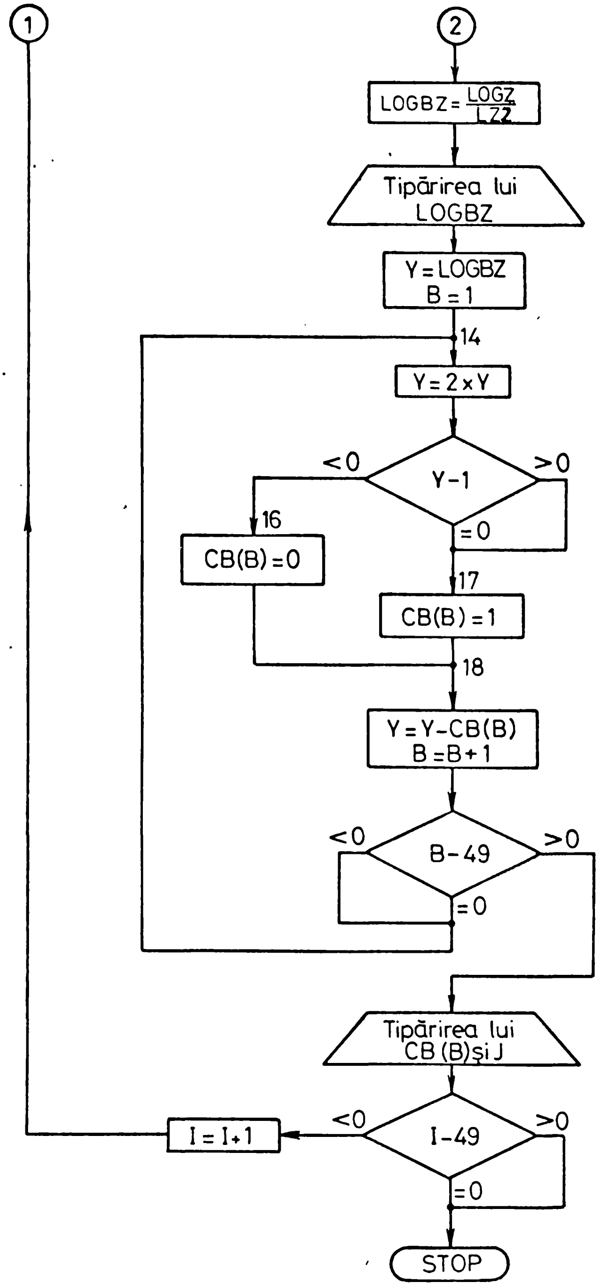


Fig.2-2 Ordinograma programului pentru calculul constantelor L_1.

BUPT

CAPITOLUL III

STRUCTURI LOGICE CELULARE PENTRU GENERAREA LOGARITMILOR ŞI ANTILOGARITMILOR BINARI.

3.1. Dispozitiv de generare a logaritmilor şi antilogaritmilor binari ce include o structură logică celulară de înmulţire şi împărţire.

Pe baza algoritmului prezentat în paragraful 2.1 din Capitolul II, elaborat de autorul tezei /66/ s-a conceput un dispozitiv de generare a logaritmilor şi antilogaritmilor binari ce include o structură logică celulară de înmulţire şi împărţire (fig. 3-1) /66/ şi utilizează în operaţii constantele C_i (Tabelul 2-1).

Dispozitivul conţine o structură logică celulară de înmulţire şi împărţire de tipul celei propusă de Gex [51], un registru al antilogaritmului (A), un registru al logaritmului (B), un şir de circuite SAU-exclusiv pentru inversarea ieşirilor structurii în cazul împărţirii [51], un generator de constante C_i precum şi unele circuite logice de control. Cu oarecare modificări se poate utiliza structura propusă de Deegan [54]. S-a adoptat însă aici structura propusă de Gex [51] deoarece aceasta are ieşiri comune pentru produs şi cît.

Cînd se calculează logaritmul binar, la început, numărul A este transferat în registrul antilogaritmului - A, se aduce la zero registrul logaritmului - B iar generatorul de constante furnizează în starea iniţială constanta $2^{1/2}$.

Impulsurile de tact I_1 deplasează spre stînga cu o poziţie conţinutul registrului B şi introduc succesiv în acesta cifrele binare ale logaritmului care sînt identice cu cifra ce apare în poziţia 2^1 a registrului A în urma operaţiilor de înmulţire sau împărţire.

Impulsurile de tact I_2 realizează înscrierea rezultatului înmulţirii sau împărţirii în registrul A şi generarea constantei următoare. Fiecare impuls de tact (fiecare ciclu) trebuie repetat de n-1 ori - n fiind numărul de cifre binare al operandului după virgulă - iar intervalul dintre două impulsuri, succesive de acelaşi tip este determinat în primul rînd de durata operaţiei de înmulţire sau împărţire [51].

Pentru stabilirea tipului operaţiei necesare la calculul logaritmului (LOG=1) se cercetează bitul b_{i-1} aflat în poziţia 2^1 a registrului A. Rezultă astfel funcţia logică I :

$$I = b_{i-1}\mathrm{LOG} \qquad (3\text{-}1)$$

BUPT

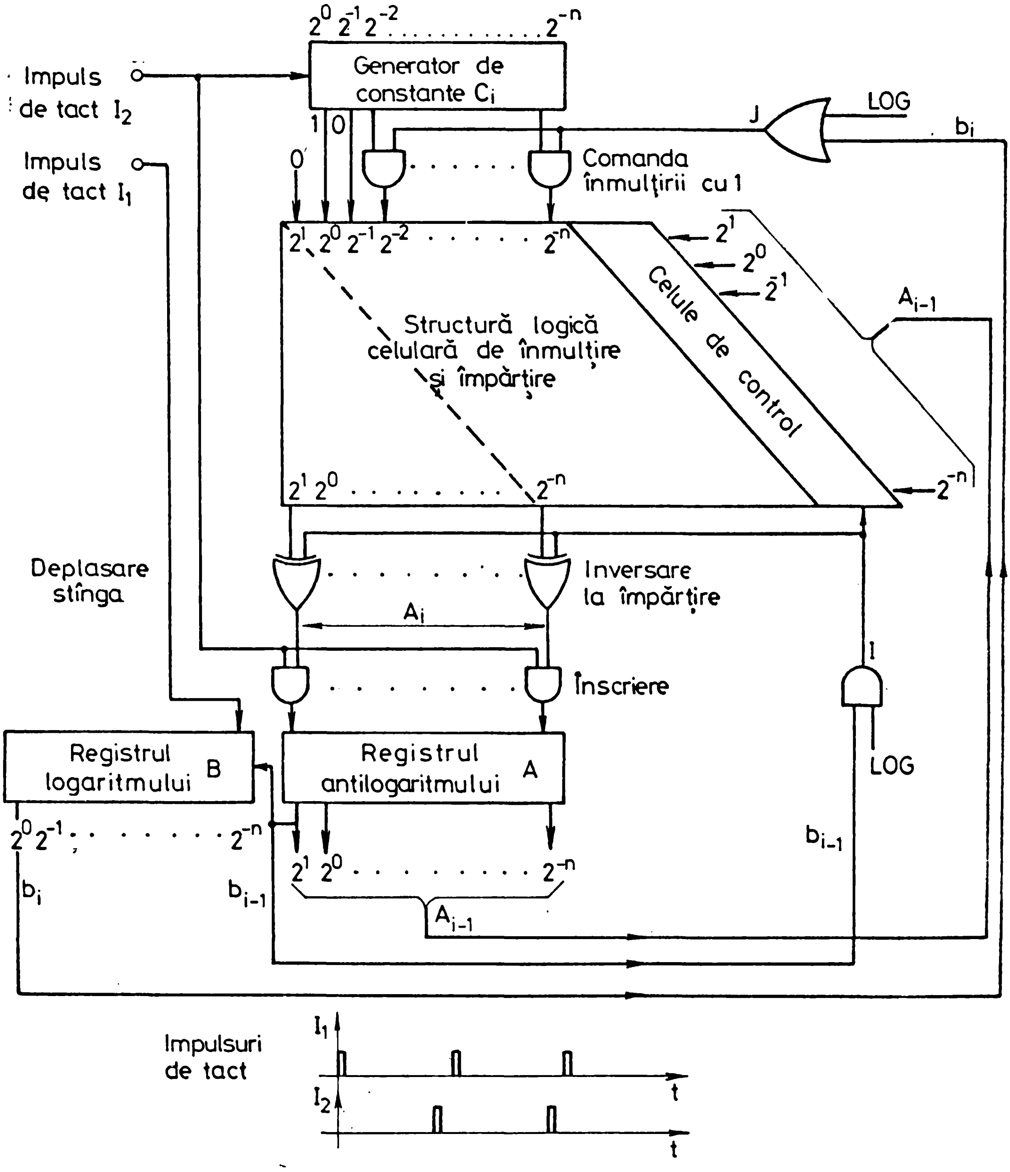


Fig.3-1. Dispozitiv de generare a logaritmilor și antilogaritmilor binari bazat pe înmulțiri și împărțiri cu constante.

Funcția I comandă prin intermediul celulelor de control [51] efectuarea operației de împărțire (cînd I=1) sau înmulțire (cînd I=0). Funcția I comandă deasemenea circuitele SAU-exclusiv pentru inversarea biților de la ieșirea structurii Gex întrucît la împărțire aceasta furnizează inversul biților cîtului [51].

BUPT

Astfel, cînd $b_{i-1} = 1$, operaţia ce trebuie efectuată este o împărţire, rezultă I=1 şi are loc inversarea ieşirilor structurii.

La calculul antilogaritmului (LOG=0) se cercetează cifra b_i a logaritmului, aflată în poziţia 2^0 a registrului B /66/, pentru a stabili dacă se face înmulţirea cu constanta $2^{1/2^i}$ sau cu 1. Rezultă astfel funcţia logică J :

$$J = LOG + b_i \qquad (3-2)$$

care la calculul antilogaritmului ia valoarea $J = b_i$. Prin intermediul circuitelor SI comandate de funcţia J se trimite de la generator la intrarea structurii constanta $2^{1/2^i}$ sau 1 după cum $b_i=1$ sau 0.

Structura de înmulţire şi împărţire Gex [51] folosită trebuie să conţină n+2 rînduri, primul rînd avînd n+2 celule iar ultimul 2n+3 celule, întrucît numărul A_{i-1} şi numărul A_i au valori cuprinse între 1 şi

$$(2-2^n).2^{1/2} = 2,824 \qquad (3-3)$$

şi au n cifre supă virgulă.

Faţă de schema propusă de Dean [19] pentru generarea logaritmilor şi antilogaritmilor binari, care necesită o structură logică celulară de ridicare la patrat (fig.1-11) şi o structură logică celulară de extragere a rădăcinii patrate (fig.1-12), schema propusă de autorul tezei (fig.3-1) conţine în plus un generator de constante cu un număr de n-1 stări, n+2 circuite SAU-exclusiv şi n circuite SI (în fig.1-11 şi 1-12 nu s-au inclus circuitele necesare pentru comanda înscrierii în registrul de lucru a rezultatului obţinut în structura logică). In schimb schema propusă aici (fig.3-1) se poate realiza simplu într-o instalaţie de calcul care conţine deja o structură logică celulară de înmulţire şi împărţire. Deasemenea, o structură logică celulară de înmulţire şi împărţire [51] conţine mai puţine circuite logice decît cele două structuri logice celulare ale schelelor Dean [19].

Timpul de generare al logaritmului realizat de schema propusă mai sus este comparabil cu cel ce se obţine la schema Dean (discutat în paragraful 1.2). El include n-1 cicluri (deoarece constanta C_n=1,000...), durata unui ciclu fiind cu ceva mai mare decît durata operaţiei de împărţire (cea mai lungă dintre operaţiile de înmulţire şi împărţire). Durata de generare a antilogaritmului va fi mai mică decît durata de generare a logaritmului deoarece se efectuează numai operaţii de înmulţire. Ea va fi, deasemenea, mult mai mică decît durata generării antilogaritmului cu circuitul propus de Dean [19] la care

BUPT

$$t_{alog} \cong 6\ t_{log} \qquad (3-4)$$

Generatorul de constante din fig.3-1 a fost studiat şi proiectat în /7o/. Se prezintă mai jos posibilităţile realizării generatorului de constante C_i, se stabileşte soluţia cea mai simplă şi se proiectează generatorul pentru un număr de 28 biţi ai constantelor după virgulă ceea ce permite utilizarea lui într-un dispozitiv aritmetic cu virgulă mobilă, ce operează cu numere avînd 24 biţi pentru mantisă.

Sînt posibile următoarele moduri de realizare a generatorului de constante /7o/ :

a) numărător binar şi matrice logică cu diode,

b) numărător cu succesiune de stări impusă,

c) registru de deplasare cu modificarea stării,

d) numărător şi memorie fixă.

In primul caz circuitele logice de comandă a numărătorului includ cîteva zeci de circuite SI cu 2 pînă la 5 intrări şi cîteva zeci de circuite SAU cu 2 pînă la 8 intrări. Aceasta face ca matricea logică să fie complicată.

In cazul al doilea funcţiile de comandă ale numărătorului sînt complicate din cauza cantităţii mari de modificări necesare la trecerea dintr-o stare în starea următoare.

In cazul al treilea se profită de forma particulară a constantelor date în Tabelul 2-1 din care se observă că după deplasarea la dreapta cu o poziţie a unei constante de ordinul i ea se deosebeşte printr-un număr redus de biţi faţă de constanta de ordinul i+1. Cantitatea de biţi ce se repetă de la o constantă la alta creşte pe măsura ce se avansează în tabel iar începînd de la mijlocul tabelului o constantă se obţine din cea anterioară prin simpla deplasare la dreapta (Tabelul 3-1). Generatorul de constante se poate deci realiza simplu, ca un registru de deplasare comandat de un impuls de tact, care se stabileşte după deplasare în starea dorită prin circuite logice simple, comandate de un alt impuls de tact (fig.3-2).

In cazul al patrulea memoria fixă poate furniza succesiv constantele C_i dacă la intrările de selecţie se aplică ieşirile de la un numărător binar obişnuit realizat cu m bistabile unde $2^m \geq n$, n fiind cantitatea de biţi ai constantelor şi în acelaşi timp cantitatea constantelor.

Proiectarea generatorului de constante de tipul a, b, sau d nu prezintă dificultăţi. Proiectarea generatorului pentru cazul c se prezintă mai jos.

INSTITUTUL POLITEHNIC TIMIŞOARA BIBLIOTECA CENTRALĂ

BUPT

TABELUL 3-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
C_{27d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C_{1}	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0
C_{1d}	1	0	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1
C_{2}	1	0	0	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1	0
C_{2d}	1	0	0	0	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1
C_{3}	1	0	0	0	1	0	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	1	1	1	0	0
C_{3d}	1	0	0	0	0	1	0	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	1	1	1	0
C_{4}	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0
C_{4d}	1	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0
C_{5}	1	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0	0	0	0	1	1	0	1	0	0	1	1
C_{5d}	1	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0	0	0	0	1	1	0	1	0	0	1
C_{6}	1	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	0	1	1	1	1	1	0
C_{6d}	1	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	0	1	1	1	1	1
C_{7}	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	0	1	0	0	1
C_{7d}	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	0	1	0	0
C_{8}	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0	1	1	1	1	1	0	1	0
C_{8d}	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0	1	1	1	1	1	0	1
C_{9}	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0
C_{9d}	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	1
C_{10}	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0	1
C_{10d}	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0
C_{11}	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	1	0	0	1	1
C_{11d}	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	1	0	0	1
C_{12}	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0	1
C_{12d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0
C_{13}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0
C_{13d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	1
C_{14}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	0
C_{14d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0
C_{15}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0
C_{15d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1
C_{16}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1
C_{16d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1
C_{17}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1
C_{17d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1
C_{18}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1
C_{18d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0
C_{19}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0
C_{19d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1
C_{20}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1
C_{20d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
C_{21}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
C_{21d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
C_{22}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
C_{22d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
C_{23}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
C_{23d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
C_{24}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
C_{24a}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
C_{25}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
C_{25d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
C_{26}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
C_{26d}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
C_{27}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

BUPT

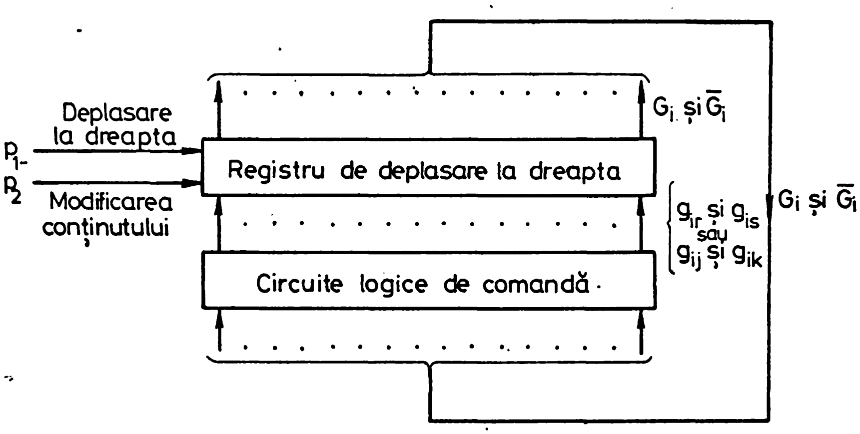


Fig.3-2. Generator de constante C_i.

Constantele C_i sînt date cu 28 biți după virgulă în Tabelul 3-1 unde cu indicele d s-au notat constantele deplasate cu o poziție spre dreapta. Comparînd două cîte două constantele, dintre care prima este deplasată, rezultă basculările necesare din starea 0 în 1 și din starea 1 în 0 a bistabilelor registrului de deplasare. Funcțiile de comandă "g" pentru bistabilele de tip RS ale generatorului rezultă deci simplu din tabel, luînd în considerație toate basculările, pe coloane :

$$
\begin{array}{ll}
g_{2s} = C_{27d} & \\
g_{3s} = C_{27d} & \qquad\qquad\qquad (3\text{-}5) \\
g_{5s} = C_{27d} & g_{5r} = C_{2d} \\
g_{6s} = C_{2d} & g_{6r} = C_{1d} \\
g_{7s} = C_{27d}+C_{2d} & \\
g_{8s} = C_{2d} & g_{8r} = C_{1d} \\
 & g_{9r} = C_{3d} \\
g_{10s} = C_{3d}+C_{1d} & \\
g_{11s} = C_{1d} & g_{11r} = C_{4d} \\
g_{12s} = C_{4d} & g_{12r} = C_{2d} \\
g_{13s} = C_{27d}+C_{1d}+C_{2d} & \\
 & g_{14r} = C_{2d}+C_{5d}+C_{6d} \\
g_{15s} = C_{1d}+C_{6d} & \\
g_{16s} = C_{27d}+C_{1d}+C_{4d}+C_{6d} & \\
g_{17s} = C_{27d} & g_{17r} = C_{4d} \\
g_{18s} = C_{27d} & g_{18r} = C_{2d}+C_{3d}+C_{4d}+C_{7d} \\
g_{19s} = C_{27d}+C_{5d} & g_{19r} = C_{2d}
\end{array}
$$

BUPT

$$g_{20r}= C_{1d}+C_{2d}+C_{8d}+C_{10d}$$

$$g_{21s}= C_{4d}+C_{6d}+C_{8d}+C_{10d}$$

$$g_{22s}= C_{3d}+C_{4d}+C_{10} \qquad g_{22r}= C_{5d}+C_{8d}+C_{9d}$$

$$g_{23s}= C_{27d}+C_{2d}+C_{3d}+C_{6d}+C_{7d}+C_{10d}\ , \qquad g_{23r}= C_{8d}+C_{4d}+C_{1d}$$

$$g_{24s}= C_{2d}+C_{5d}+C_{10d} \qquad g_{24r}= C_{1d}+C_{3d}+C_{6d}$$

$$g_{25s}= C_{1d}+C_{2d}+C_{7d} \qquad g_{25r}= C_{8d}+C_{11d}$$

$$g_{26s}= C_{27d}+C_{5d}+C_{9d}+C_{11d} \qquad g_{26r}= C_{4d}+C_{6d}+C_{7d}$$

$$g_{27s}= C_{27d}+C_{5d}+C_{7d}+C_{8d} \qquad g_{27r}= C_{9d}+C_{6d}+C_{3d}$$

$$g_{28s}= C_{4d}+C_{10d} \qquad g_{28r}= C_{1d}+C_{2d}+C_{5d}+C_{8d}+C_{13d}$$

Dacă bistabilele sînt de tip JK atunci expresiile funcţiilor de comandă rămîn aceleaşi şi se înlocuiesc doar indicii r cu j şi s cu k. Această echivalare este valabilă deoarece funcţiile g_{ir} şi g_{is} nu sînt niciodată simultan egale cu 1.

In aceste funcţii logice intră ca variabile numai stările C_{id} ale registrului după deplasare pentru i=27 şi i= 1...13. In celelalte situaţii nu este necesară comanda modificării stării registrului. Expresiile C_{id} pentru stări se pot deduce simplu din Tabelul 3-1 incluzînd în ele mai întîi bitul G_{i+1} (corespunzător cifrei 1 din poziţia $2^{-(i+1)}$ care deosebeşte o stare C_{id} de toate stările următoare din tabel şi încă unul sau mai mulţi biţi, negaţi sau nu, care deosebesc starea C_{id} de toate stările anterioare din tabel care au bitul $G_{i+1}= 1$. In felul acesta expresiile pentru stările C_{id} rezultă foarte simplu :

$$C_{1d} = G_3 \qquad C_{2d} = G_4G_5 \qquad C_{3d} = G_5G_7 \qquad (3\text{-}6)$$

$$C_{4d} = G_6G_9 \qquad C_{5d} = G_7\bar{G}_8 \qquad C_{6d} = G_8G_{10}$$

$$C_{7d} = G_9G_{19} \qquad C_{8d} = G_{10}G_{12} \qquad C_{9d} = G_{11}\bar{G}_{17}$$

$$C_{10d}= G_{12}\bar{G}_{17} \qquad C_{11d}= G_{13}G_{15}G_{16} \qquad C_{12d}= G_{14}G_{21}$$

$$C_{13d}= G_{15}\bar{G}_{16}G_{17} \qquad C_{27d}= \bar{G}_{25}\bar{G}_{26}\bar{G}_{27}\bar{G}_{28}$$

Printr-o metodă asemănătoare cu cea de scriere a stărilor C_{id} se poate efectua minimizarea expresiilor funcţiilor de comandă g_{is} şi g_{ir} de mai sus care includ funcţii SAU de stări. Se poate găsi un număr redus de biţi G_j, comuni tuturor stărilor din funcţia SAU ce se minimizează şi care nu apar împreună în nici una din celelalte stări ale registrului.

In urma acestei minimizări rezultă funcţiile de comandă în forma finală :

$$g_{2s}= \bar{G}_{25}\bar{G}_{26}\bar{G}_{27}\bar{G}_{28} \qquad g_{3s} = g_{2s} \qquad (3\text{-}7)$$

$$g_{5s}= g_{2s} \qquad g_{5r} = G_4G_5$$

$$g_{6s}= G_3 \qquad g_{7s} = g_{2s}+g_{5r}$$

BUPT

$g_{8s} = g_{5r}$

$g_{9r} = G_5 G_7$

$g_{11s} = G_3$

$g_{12s} = g_{11r}$

$g_{13s} = g_{2s} + G_4$

$g_{15s} = G_8 \overline{G}_9$

$g_{17s} = g_{2s}$

$g_{18s} = g_{2s}$

$g_{19r} = g_{5r}$

$g_{21s} = G_8 G_9 + G_{12} \overline{G}_{17}$

$g_{23s} = G_{13} \overline{G}_{21} \overline{G}_{22} \overline{G}_{23} + \overline{G}_{25} \overline{G}_{27} \overline{G}_{28}$

$g_{24s} = G_{14} G_{16} \overline{G}_{24} + G_{12} \overline{G}_{17}$

$g_{25s} = G_{17} G_{18} G_{19}$

$g_{26s} = g_{2s} + G_{13} \overline{G}_{26}$

$g_{27s} = g_{2s} + \overline{G}_{15} G_{17} G_{22}$

$g_{28s} = G_{15} \overline{G}_{16} \overline{G}_{25}$

$g_{8r} = G_3$

$g_{10s} = G_8 G_{10} \overline{G}_{11}$

$g_{11r} = G_6 G_9$

$g_{12r} = g_{5r}$

$g_{14r} = G_4 \overline{G}_6 G_{17} G_{21} \overline{G}_{28}$

$g_{16s} = g_{2s} + G_8 \overline{G}_{12}$

$g_{17r} = g_{11r}$

$g_{18r} = g_{5r} + G_2 \overline{G}_{10}$

$g_{20r} = G_5 + G_{12} \overline{G}_{16}$

$g_{22r} = G_{13} \overline{G}_{15}$

$g_{23r} = G_{17} G_{18} G_{23}$

$g_{24r} = \overline{G}_{15} G_{17} G_{18} \overline{G}_{21}$

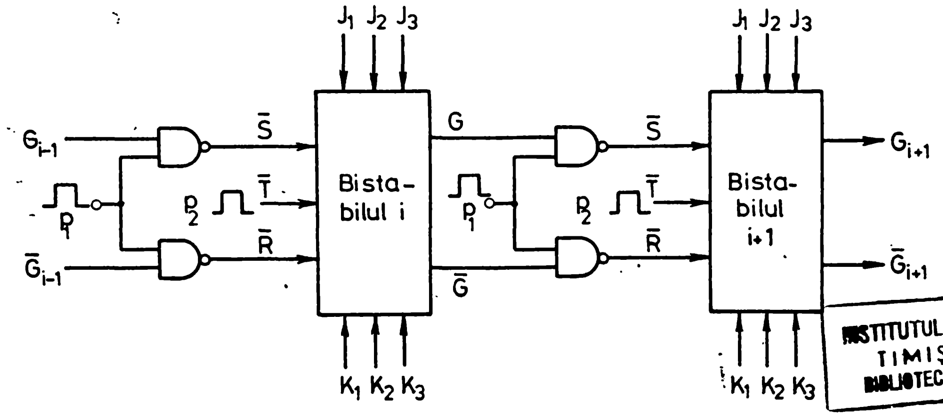
$g_{25r} = G_{13} G_{20}$

$g_{26r} = G_{11} G_{24}$

$g_{27r} = G_{14} G_{18} \overline{G}_{23} G_{27}$

$g_{28r} = G_{17} \overline{G}_{19} G_{22} + G_4$

Circuitele de comandă ale registrului de deplasare din generatorul de constante C_i sînt cele mai simple atunci cînd registrul este realizat cu bistabile integrate master-slave JK (fig.3-3) avînd cîte 3 intrări J şi K ce formează o funcţie logică SI.



INSTITUTUL POLITEHNIC
TIMIŞOARA
BIBLIOTECA CENTRALĂ

Fig.3-3. Realizarea registrului de deplasare al generatorului de constante.

In acest caz o serie de funcţii logice din lista de mai sus şi anume, cele de forma SI, se realizează direct de către intrările bistabilelor. Deplasarea la dreapta se face în prezenţa impulsului p_1 prin intermediul intrărilor asincrone iar modificarea conţinutului se face prin înscrierea în bistabilul master cu ajuto-

BUPT

rul funcţiilor de comandă în timpul impulsului p_2 apoi prin înscrierea în bistabilul slave la terminarea impulsului p_2 care ţine locul impulsului de orologiu (aplicat la intrarea de sincronizare).

La realizarea generatorului sînt necesare următoarele cantităţi de circuite : 27 bistabile master-slave JK, 4 invertoare, 72 circuite NU-SI cu 2 intrări, 4 circuite NU-SI cu 3 intrări, 4 circuite NU-SI cu 4 intrări, 1 circuit NU-SI cu 8 intrări. Acestea reprezintă 51 capsule integrate din care pentru realizarea funcţiilor de comandă sînt utilizate doar 11 capsule /7o/.

In mod asemănător se poate proiecta şi un generator pentru constantele L_i utilizate (paragraful 2.2) în cadrul algoritmului bazat pe descompunere în factori /7o/ dacă acesta se implementează pe un dispozitiv aritmetic secvenţial obişnuit.

3.2. Structură logică celulară de generare în paralel a antilogaritmilor binari bazată pe înmulţirea constantelor C_i.

Pe baza algoritmului de calcul al antilogaritmilor binari prezentat în paragraful 2-1 (relaţia 2-5) s-a conceput o structură logică celulară de generare în paralel a antilogaritmului binar /56/. Plecînd de la relaţia :

$$A = 2^{\frac{1}{2^1}b_1}\ 2^{\frac{1}{4}b_2} \dots 2^{\frac{1}{2^i}b_i} \qquad (3\text{-}8)$$

unde b_i reprezintă bitul de ordinul i al logaritmului, şi făcînd substituţia :

$$2^{\frac{1}{2^i}b_i} = \left[2^{\frac{1}{2^i}}\right]^{b_i} = (C_i)^{b_i} \qquad (3\text{-}9)$$

se obţine :

$$A = (C_1)^{b_1}(C_2)^{b_2}\dots(C_i)^{b_i} = \prod_{i=1}^{i=n}(C_i)^{b_i} \qquad (3\text{-}1o)$$

Un algoritm pentru operaţia de calcul al logaritmului care să conţină numai operaţii de înmulţire cu constante nu a fost găsit de autorul tezei (algoritmul de calcul al logaritmului prezentat în paragraful 2-1 include operaţii de înmulţire şi împărţire).

Inseamnă, aşa cum s-a arătat şi în paragraful 2-1, că se poate calcula antilogaritmul binar al unui număr binar L prin înmulţirea între ele a acelor constante C_i (Tabelul 2-1) care corespund biţilor b_i diferiţi de zero ai numărului L. Cantitatea înmulţirilor nu va fi totuşi exageratădeoarece nu întotdeauna toţi biţii numărului binar L sînt 1-uri. Deasemenea înmulţirea cu o constantă este mai uşor de efectuat întrucît se pot lua numai

BUPT

acele produse parțiale (deplasate în modul necesar) care corespund biților egali cu 1 ai constantelor C_i (Tabelul 2-1).

Pentru alcătuirea structurii logice celulare de antilogaritmare relația (3-10) se pune în forma :

$$A = \prod_{i=1}^{i=n-1} \left[1+b_i(C_i-1)\right] \qquad (3\text{-}11)$$

unde s-a folosit substituția :

$$(C_i)^{b_i} = 1+b_i(C_i-1), \qquad (3\text{-}12)$$

pentru a se lucra numai cu partea fracționară a constantelor C_i și s-a ținut cont că $C_n=1$ (Tabelul 2-1).

Dacă se notează produsul parțial de ordinul i cu P_i atunci se poate scrie /56/ :

$$P_i = P_{i-1}\left[1+b_i(C_i-1)\right] = P_{i-1}b_i(C_i-1) =$$

$$= P_{i-1}+P_{i-1}b_i(c_{i1}2^{-1}+c_{i2}2^{-2}+ \dots + c_{in}2^{-n}) \qquad (3\text{-}13)$$

unde $c_{i1}, c_{i2}, \dots c_{in}$ reprezintă biții din dreapta virgulei ai a constantei C_i ($c_o=1$ întotdeauna).

Prin urmare produsul parțial rezultat după cea de a i-a înmulțire se obține adăugînd la produsul parțial anterior, cînd $b_i=1$, același produs, repetat de atîtea ori cîți biți 1 conține constanta C_i și deplasat în conformitate cu ordinul acestora. Cînd $b_i=0$ la produsul parțial anterior nu se mai adaugă nimic. Produsul parțial inițial va fi :

$$P_1 = 1 + b_1\,(C_1 - 1) \qquad (3\text{-}14)$$

urmînd ca el să fie înmulțit cu (n-2) constante de forma (3-12).

Structura logică celulară care efectuează operația de antilogaritmare după relațiile (3-11) și (3-13) conține n-2 structuri logice celulare de multiplicare (fig.3-5) de tipul celei descrise în [8] care au însă fiecare numai acele rînduri ce corespund biților egali cu 1 ai părții fracționare a constantelor C_i.

Structurile de multiplicare se pot realiza cu celula propusă tot în [8] și prezentată în fig.3-4. Aici cu indicele i s-a notat ordinul bitului constantei C_i-1.

In celulă se realizează funcțiile logice :

$$P_{k,j} = (P_{k-1,j} \oplus P_{i-1,j-k} \oplus T_{k,j})b_i c_{i,k} + P_{k-1,j}\overline{b_i c_{i,k}} \qquad (3\text{-}15)$$

$$T_{k,j-1} = P_{k-1,j}P_{i-1,j-k} + P_{k-1,j}T_{k,j} + P_{i-1,j-k}T_{k,j} \qquad (3\text{-}16)$$

BUPT

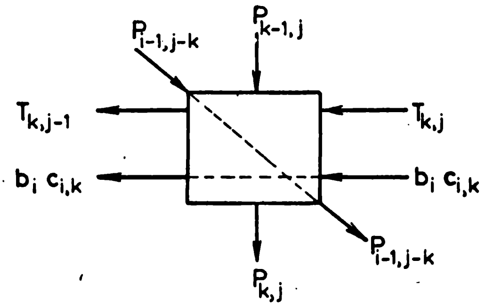


Fig.3-4. Celula structurii de generare a antilogaritmului.

Deci ieşirea $P_{k,j}$ reprezintă suma logică a biţilor $P_{k-1,j}$, $P_{i-1,j-k}$ şi $T_{k,j}$, condiţionată de valoarea bitului b_i (conform relaţiei (3-13). Bitul $c_{i,k}$ este în totdeauna egal cu 1 întrucît structura de multiplicare conţine numai rîndurile corespunzătoare cifrelor înmulţitorului $b_i(C_i-1)$ care sînt egale cu 1.

Ieşirea $T_{k,j-1}$, reprezintă transportul rezultat în cadrul însumării biţilor amintiţi.

La intrările verticale şi diagonale ale structurii de multiplicare de ordinul i (fig.3-5) se aplică produsul parţial P_{i-1} cu lungime dublă (cu2n biţi după virgulă), respectiv simplă iar la intrările orizontale produsul :

$$b_i(C_i-1) = b_i(c_{i1}2^{-1}+ c_{i2}2^{-2}+ \dots c_{in}2^{-n}) , \qquad (3-17)$$

Prima structură de multiplicare nu mai este necesară deoarece primul factor, $1+b_1(C_1+1)$ se aplică la intrările verticale ale structurii de multiplicare 2.

Cu linie întreruptă s-a marcat limita structurii de multiplicare propusă de Dean [8] pentru cazul cînd P_{i-1} adus la intrare are numai n biţi. Aici însă structura trebuie completată ca în fig.3-5 pentru a se putea utiliza produsul P_{i-1} cu lungimea dublă în scopul reducerii erorilor de rotunjire.

Săgeţile arată sensul în care creşte ordinul biţilor. Coloana corespunzătoare bitului dinaintea virgulei se poate elimina deoarece acest bit este în totdeauna egal cu 1 şi nici nu apare un transport din coloana din dreapta virgulei ($1 \leq A<2$). In acest caz la prima intrare diagonală din stînga a fiecărei structuri de multiplicare trebuie aplicat bitul 1. La intrările diagonale neutilizate ale celulelor trebuie aplicat bitul 0 (în triunghiul delimitat de linia întreruptă).

Din Tabelul 2-1 se vede că lucrînd cu lungime de cuvînt n=28 biţi după virgulă trebuie luate în considerare 27 constante C_i (C_{28}= 1). Structura logică celulară de antilogaritmare va avea atunci un număr de 26 structuri de multiplicare incomplete avînd împreună 184 rînduri (numărul de 1-uri conţinut de partea fracţionară a constantelor $C_2 \dots C_{27}$)şi un total de celule identice de

BUPT

8348 (cînd coloana din stînga virgulei este eliminată). Numărul celulelor şi rîndurilor scade rapid la reducerea numărului de biţi ai operandului. Astfel, pentru n=23 rezultă 131 rînduri şi 443o celule iar pentru n=18-85 de rînduri cu 2227 celule.

Dacă în schema din fig.3-5 se folosesc structuri de multiplicare avînd la intrare produsul parţial cu lungime simplă [14] atunci celulele din colţul delimitat prin linie întreruptă se pot înlătura, micşorîndu-se astfel numărul total de celule din structură dar crescînd totodată eroarea de calcul cu un ordin binar (Capitolul II). In acest caz numărul de rînduri ale structurii rămîne acelaşi iar cantitatea celulelor este :

- pentru n=28 : N_c= 5152
- pentru n=23 : N_c= 3013
- pentru n=18 : N_c= 153o

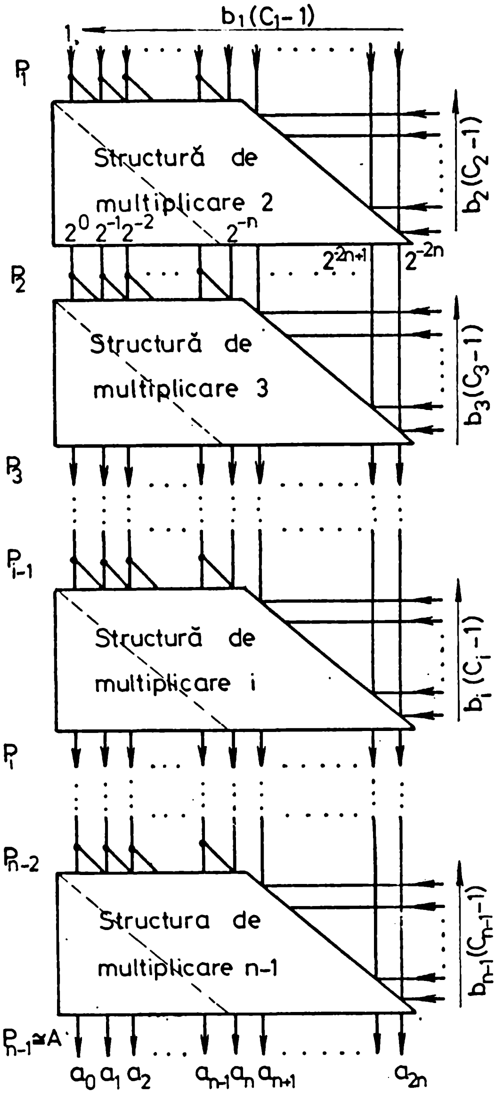


Fig.3-5. Structura logică celulară de generare a antilogaritmului.

In cazul cel mai defavorabil, cînd toţi biţii b_i sînt egali cu 1 operaţia de antilogaritmare durează un timp ce poate fi delimitat superior prin timpul de propagare al transportului de lungime maximă posibilă. Acesta trece pe diagonală peste un număr de celule egal cu N_r-(n-2) unde N_r este numărul de rînduri al structurii complete iar pe orizontală peste cel mult :

$$n+(n-1)+(n-2)+\ldots+1=\frac{n(n+1)}{2} \qquad (3\text{-}18)$$

dacă se consideră că în fiecare structură logică de multiplicare prin propagarea transportului se formează cîte un bit 1 al antilogaritmului (antilog$_2$ o,1111... = 1,1111....).

BUPT

Timpul de propagare pe diagonală şi pe orizontală al transportului se poate considera acelaşi deoarece el trece peste un singur nivel logic (pe diagonală apar circuite invertoare pentru amplificare întrucît în fiecare celulă intrarea diagonală comandă 6 intrări de circuite logice). Astfel, durata totală a procesului de antilogaritmare va fi :

$$t_{alog} < \left[N_r-(n-2)+ \frac{n(n+1)}{2}\right]t_p = \left[N_r+ \frac{n(n-1)}{2} +2 \right] t_p \quad (3\text{-}19)$$

Pentru n=28 acest timp devine $t_{alog} < 564\ t_p$. Dacă se consideră $t_p = 6$ ns atunci rezultă $t_{alog} < 3384$ ns $\cong 3{,}4\ \mu s$. Această durată este mult mai mică decît cea realizată cu schema serie-paralel propusă de autorul tezei /66/, comentată în paragraful 3-1 (fig. 3-1) unde $t_{alog} \cong 40\ \mu s$ (relaţia 1-29) şi deasemenea mult mai mică decît durata realizată de dispozitivul propus de Dean [19] (relaţia 1-32) : $t_{alog} < 215\ \mu s$. Prin urmare un avantaj important al acestei structuri logice celulare de antilogaritmare îl constituie timpul de calcul redus.

Un alt avantaj constă în faptul că nu se utilizează în schemă registre iar comenzile externe se limitează doar la aplicarea la intrări a biţilor b_i.

Totuşi cantitatea celulelor din structura logică propusă (fig.3-1), pentru $n \geqslant 20$ este exagerat de mare şi ea poate interesa numai în cazul folosirii integrării pe scară largă.

Deasemenea, lipsa unui algoritm asemănător pentru calculul logaritmilor binari constituie un dezavantaj important al acestei metode de calcul a antilogaritmilor. Pe baza celor arătate mai sus se poate trage concluzia că această structură este utilizabilă numai în unele instalaţii de calcul specializate.

3.3. Structuri logice celulare de generare a logaritmilor şi antilogaritmilor binari bazate pe descompunerea în factori şi termeni.

In paragraful 2.2. s-a prezentat algoritmul de calcul a logaritmului binar bazat pe descompunerea în factori de forma $1+k_i 2^{-i}$ a numărului ce se logaritmează şi pe însumarea logaritmilor factorilor (relaţiile 2-34 şi 2-35). Calculul antilogaritmului se face prin descompunerea în termeni de forma $\log_2(1+2^{-i})$ a numărului ce se antilogaritmează şi prin înmulţirea antilogaritmilor termenilor.

In cele ce urmează se vor prezenta structuri logice celulare care, utilizînd algoritmul prezentat anterior /58/, generează logaritmii sau antilogaritmii binari.

BUPT

Circuitele au fost concepute pentru numere ce se logaritmează cuprinse în domeniul $1 \leq A < 2$ şi pentru numere ce se antilogaritmează cuprinse în domeniul $0 \leq L < 1$.

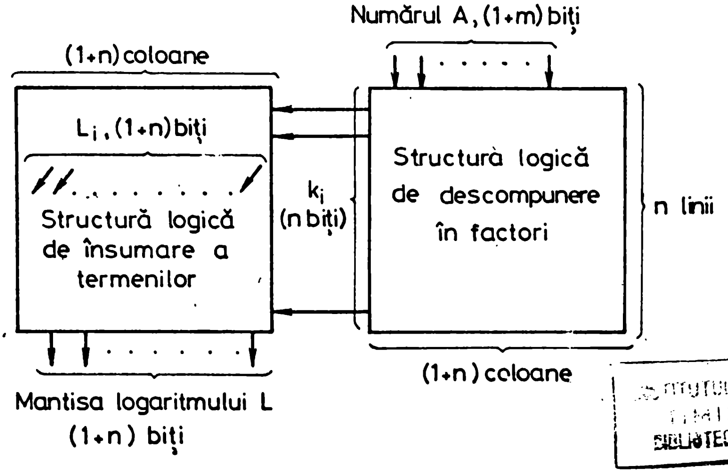
In acest fel circuitele propuse pot furniza mantisa logaritmului respectiv forma fracţionară (cu virgula după prima cifră 1) a antilogaritmului, fiind utilizabile într-un dispozitiv aritmetic cu virgulă mobilă. Caracteristica logaritmului respectiv cantitatea cifrelor întregi ale antilogaritmului se determină în acest caz uşor din exponenţii numerelor reprezentate în sistemul cu virgulă mobilă.

3.3.1. Structuri logice celulare pentru generarea logaritmilor binari /37/.

Calculul logaritmului binar al numărului A, prezentat în paragraful 2.2 include următoarele operaţii aritmetice :

- descompunerea numărului A în factori de forma $1+k_i 2^{-i}$ (determinarea coeficienţilor de creştere k_i),
- însumarea logaritmilor factorilor, adică însumarea termenilor L_i corespunzători valorilor k_i egale cu 1.

Aceste două operaţii se pot efectua în paralel cu ajutorul circuitelor din fig.3-6.



INSTITUTUL POLITEHNIC TIMIŞOARA BIBLIOTECA CENTRALĂ

Fig.3-6. Structuri logice celulare pentru generarea logaritmilor binari.

In fig.3-7 şi fig.3-10 se prezintă structurile logice celulare de descompunere în factori respectiv - de însumare a termenilor pentru o cantitate de biţi n=9.

Calculînd eroarea absolută posibilă la calculul logaritmu-

BUPT

lui cu relaţia (2-54) rezultă :

$$\Delta L_p = (2n-1)2^{-(n+1)} = 17.2^{-10} = 2^{-6}+2^{-10} \qquad (3\text{-}20)$$

Pentru ca eroarea să nu afecteze prea mult calcul cu logaritmi structurile pentru n=9 biţi din fig.3-7 şi 3-10 se pot folosi într-o instalaţie de calcul ce lucrează cu numere avînd n=6 biţi după virgulă. In cele ce urmează se consideră că numărul A este adus la intrarea structurii logice de descompunere în factori cu m biţi după virgulă (aşa cum se găseşte el în calculator m<n) iar în interiorul structurilor se lucrează cu n biţi. Logaritmul este furnizat deci cu n biţi dar cu o eroare de ordinul 2^{-m}. Din aceştia se pot folosi în calcule fie numai m biţi, fie toţi n biţii.

Fiecare din cele două structuri logice este compusă din celule identice avînd schema logică din fig.3-8 şi 3-11.

Structura logică de descompunere în factori.

Intr-un rînd i al structurii logice de descompunere în factori se efectuează următoarele operaţii :

- înmulţirea produsului anterior

$$P_{i-1} = \prod_{i=1}^{i=i-1} (1+k_i 2^{-i}) \qquad (3\text{-}21)$$

(produsul factorilor incluşi în dezvoltare pînă la acel rînd) cu factorul corespunzător rîndului în cauză - $(1+2^{-i})$,

- compararea produsului nou obţinut cu numărul A ce se descompune. Compararea se face prin determinarea cifrei împrumutului la scădere în fiecare celulă şi propagarea acestuia spre ultima celulă din stînga ; împrumutul rezultat în rangul cel mai semnificativ va reprezenta negaţia coeficientului de creştere k_i,

- transmiterea spre rîndul următor (i+1) a produsului de la intrare P_{i-1} nemodificat cînd k_i=0 ($\bar{k}_i$=1) sau a produsului nou $P_{i-1}\cdot(1+2^{-i})$ cînd k_i=1 ($\bar{k}_i$=0).

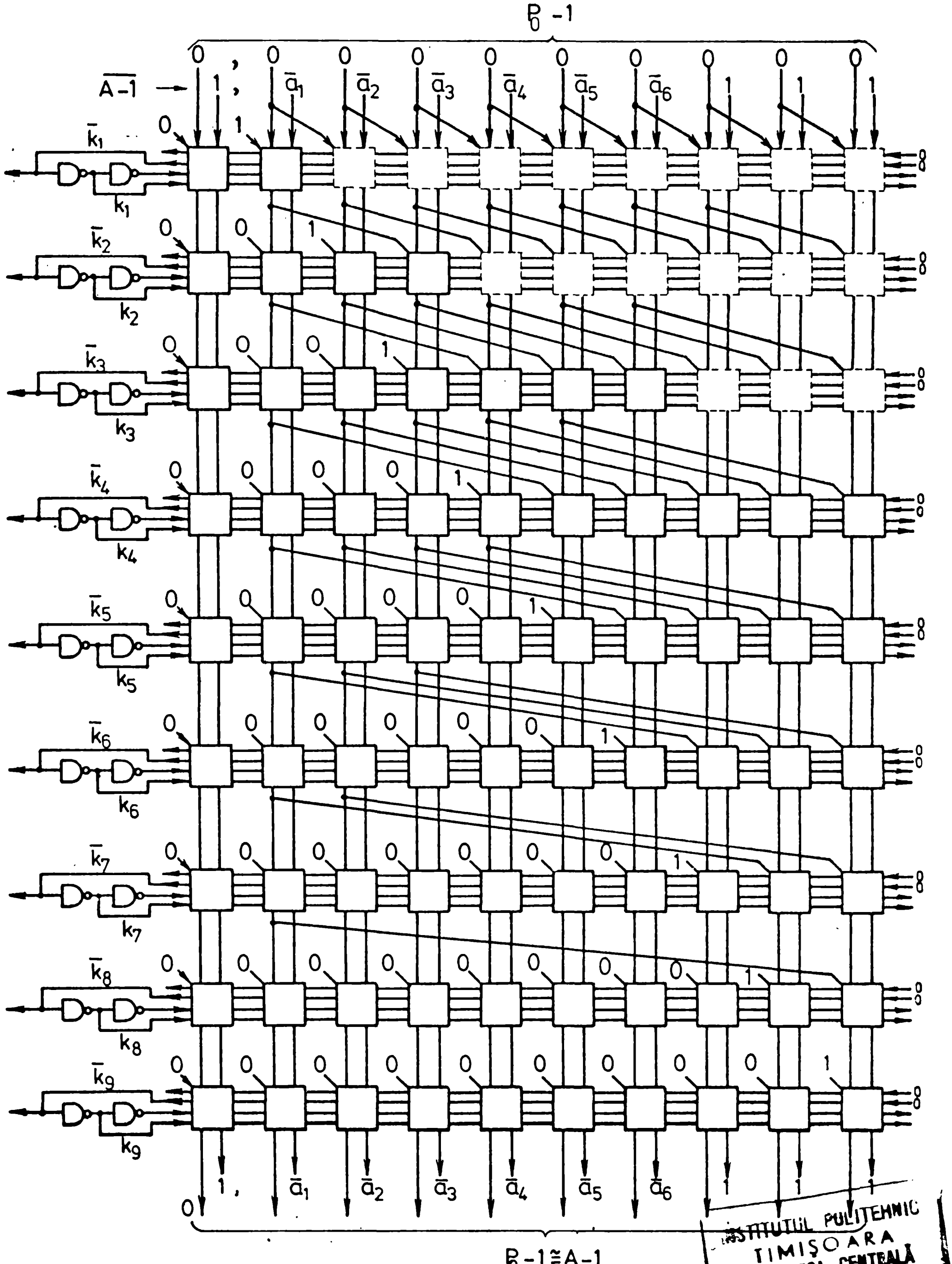
Prin urmare în rîndul i se efectuează înmulţirea :

$$P_{i-1}(1+2^{-i}) = P_{i-1} + 2^{-i}P_{i-1} \qquad (3\text{-}22)$$

care constă din adunarea produsului anterior P_{i-1} cu el însuşi deplasat cu i poziţii binare spre dreapta.

Deplasarea cu i poziţii spre dreapta este realizată în structura logică din fig.3-7 prin conexiunile oblice dintre două rînduri iar însumarea se face în interiorul celulelor de tipul celei din fig.3-8. Ca produs iniţial trebuie luat numărul P_0=1,000... iar numărul A ce se descompune trebuie completat cu zerouri pînă la n biţi

BUPT



INSTITUTUL POLITEHNIC
TIMIŞOARA
BIBLIOTECA CENTRALĂ

Fig.3-7. Structura logică celulară de descompunere în factori.

BUPT

după virgulă (în structura din fig.3-7 se utilizează la intrare negațiile biților numărului A). In cazul unei descompuneri exacte cifrele numărului A și ale produsului P_n de la ieșirea structurii (după rîndul n) trebuie să coincidă.

In structura logică de descompunere în factori în poziția 2^1 (în stînga primei coloane din fig.3-7) ar fi necesară încă o coloană deoarece produsul $P_{i-1}(1+2^{-i})$ poate deveni mai mare decît 2. Astfel, dacă se face produsul tuturor factorilor (așa cum s-a arătat în capitolul 2) rezultă :

$$\prod_{i=1}^{i=n} (1+2^{-i}) = 2,38 \qquad (3\text{-}23)$$

Desigur, în urma comparării cu numărul A, rezultă $k_i=0$ și un produs mai mare decît 2 nu este transmis spre rîndul următor. Pentru a se elimina totuși, această coloană din poziția 2^1 se va lucra cu numerele A-1 și $P_{i-1}-1$, cuprinse în domeniile :

$$\begin{aligned} 0 &\leq A-1 < 1 \\ 0 &\leq P_{i-1}-1 < 2 \end{aligned} \qquad (3\text{-}24)$$

Rezultă astfel suma :

$$s = P_{i-1}(1+2^{-i}) - 1 = (P_{i-1}-1)+(P_{i-1}-1).2^{-i}+ 2^{-i} \qquad (3\text{-}25)$$

Cu alte cuvinte, în structura logică se va lucra numai cu partea funcționară a numărului A și a lui P_{i-1}. Scăderea unui 1 din produsul P_{i-1} se face prin scăderea lui 1 din produsul inițial P_0 de la intrarea structurii. Din relația (3-25) se vede modul în care se adună produsul anterior deplasat :

$$2^{-i}P_{i-1} = 2^{-i}(P_{i-1}-1) + 2^{-i} \qquad (3\text{-}26)$$

adică se adună partea fracționară a lui P_{i-1} deplasată cu i poziții la dreapta iar în coloana i după virgulă se adună un 1 (fig.3-7). Aceste mărimi se aplică la intrările diagonale ale celulei (fig.3-8).

La intrările diagonale unde nu se aduce produsul anterior deplasat trebuie aplicate cifre 0.

Comparația între A și $P_{i-1}(1+2^{-i})$ se face prin determinarea împrumutului la scăderea :

$$(A-1)-s = (A-1)-\left[P_{i-1}(1+2^{-i})-1\right] = A-P_{i-1}(1+2^{-i}) \qquad (3\text{-}27)$$

Utilizarea unor circuite de comparare rapidă [46] ar duce la o creștere a vitezei de calcul dar din cauza dimensiunilor lor ele merită a fi luate în considerare doar în momentul cînd vor fi disponibile în forma de circuite integrate pe scară largă.

Ieșirea de împrumut de la celula din stînga ($\overline{k}_i$) a fiecărui

BUPT

rînd şi negaţia ei ($\bar{k}_i$) se retrimit prin toate celulele rîndului respectiv. Se controlează astfel transmiterea spre rîndul următor a părţii fracţionare a produsului corect al factorilor, adică a produsului anterior P_{i-1}, cînd k_i=0 sau a produsului nou :

$$P_i = P_{i-1}(1+2^{-i}) \qquad (3\text{-}28)$$

cînd k_i= 1.

In cadrul celulei din rîndul i, coloana j, utilizată în structura logică de descompunere în factori (fig.3-8) se realizează următoarele funcţii logice pe baza variabilelor de intrare:

$$s_{i,j} = P_{i-1,j} \oplus P_{i-1,j-1} \oplus T_{i,j+1} \qquad (3\text{-}29)$$

$$T_{i,j} = P_{i-1,j}\, P_{i-1,j-i} + P_{i-1,j}\, T_{i,j+1} + P_{i-1,j-i}\, T_{i,j+1} \qquad (3\text{-}30)$$

$$I_{i,j} = \bar{A}_j(s_{i,j} + I_{i,j+1}) + s_{i,j} I_{i,j+1} \qquad (3\text{-}31)$$

$$P_{i,j} = P_{i-1,j}\bar{k}_i + s_{i,j} k_i \qquad (3\text{-}32)$$

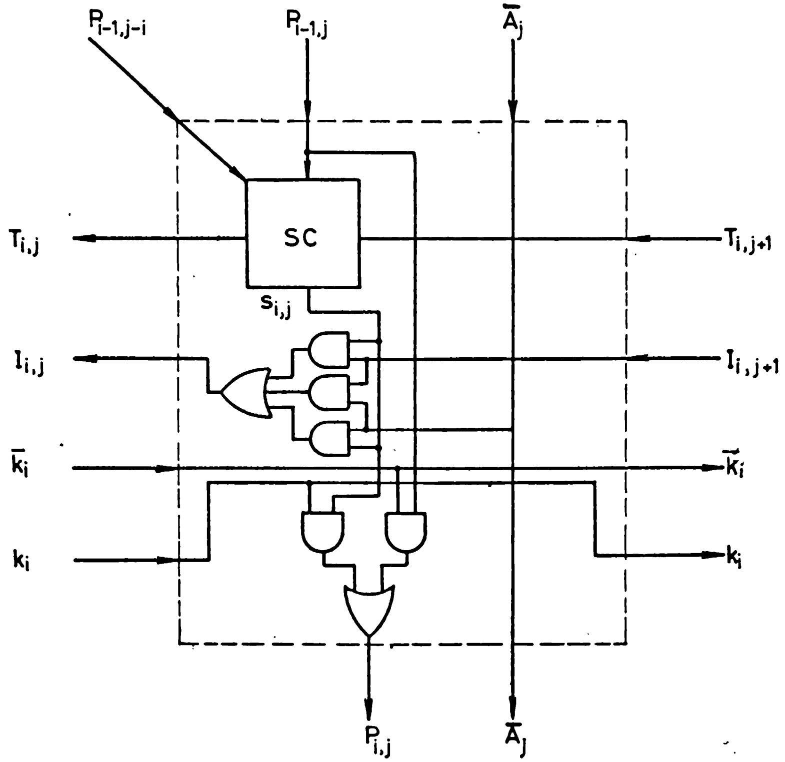


Fig.3-8. Celula structurii de descompunere în factori.

BUPT

Funcţia $s_{i,j}$ reprezintă suma logică a biţilor $P_{i-1,j}$; $P_{i-1,j-1}$; $T_{i,j+1}$ iar funcţia logică $T_{i,j}$ reprezintă transportul rezultat la adunarea acestora în sumatorul complet SC. $T_{i,j+1}$ reprezintă transportul din rangul anterior.

Funcţia logică $I_{i,j}$ reprezintă împrumutul rezultat la scăderea din bitul A_j a biţilor $s_{i,j}$ şi $I_{i,j+1}$ (împrumutul din rangul anterior).

Funcţia logică $P_{i,j}$ reprezintă bitul produsului corect, fiind egală cu P_{i-1}, cînd $k_i=0$ sau cu $s_{i,j}$ cînd $k_i=1$.

Celulele (fig.3-8) reclamă prezenţa negaţiilor biţilor A_j la intrarea structurii. Deasemenea, pentru simplitatea celulei (prin eliminarea unor invertoare suplimentare ce măresc timpul de propagare) este necesar să se utilizeze două bare pentru k_i şi $\bar{k}_i$ care să treacă prin toate celulele unui rînd. Barele $\bar{A}_j, k_i$ şi $\bar{k}_i$ acţionează cîte două, respectiv o intrare de circuit logic în fiecare celulă de aceea numărul total de intrări din structura logică pe care le acţionează aceste bare este egal cu 2n pentru $\bar{A}_j$ şi n+1 pentru k_i şi $\bar{k}_i$. Există circuite integrate NU-SI de putere [69] care pot acţiona un număr de 3o intrări de circuite logice. Se pot utiliza deci astfel de circuite pentru comanda barelor $\bar{A}_j$, k_i, $\bar{k}_i$ din structura logică celulară de mai sus.

Din relaţia 3-25 se vede că în fiecare rînd al structurii logice de descompunere în factori cantitatea de biţi a sumei dintr-un rînd creşte cu i biţi spre dreapta datorită adunării produsului deplasat $2^{-i}P_{i-1}$. Deoarece în primul rînd al structurii produsul anterior este $P_0= 1,000...$ rezultă $s_1= 2^{-1}$ adică apare un singur bit în dreapta virgulei iar în continuare, în fiecare rînd, cantitatea de biţi ai numărului s_i creşte cu i faţă de cea a lui s_{i-1}. Deci în celulele din colţul din dreapta de sus al structurii (cu linie întreruptă în fig.3-7) nu se efectuează operaţii aritmetice şi aceste celule nu sînt necesare în structură.

Pentru un număr n de biţi dat structura are în acest caz conturul din fig.3-9. Numărul total de celule utilizate va fi în acest caz :

$$N_c= (1+1)+(1+1+2)+(1+1+2+3)+...+(1+1+2+3+...+s)+(1+n)(n-s) =$$

$$= s + \sum_{i=1}^{i=s} i(s+1-i)+(1+n)(n-s) \qquad (3-33)$$

unde s se determină din condiţia :

$$1+2+3+...+s \leq n \qquad (3-34)$$

Astfel, pentru n = 28 rezultă s = 7 şi N_c = 7oo.

BUPT

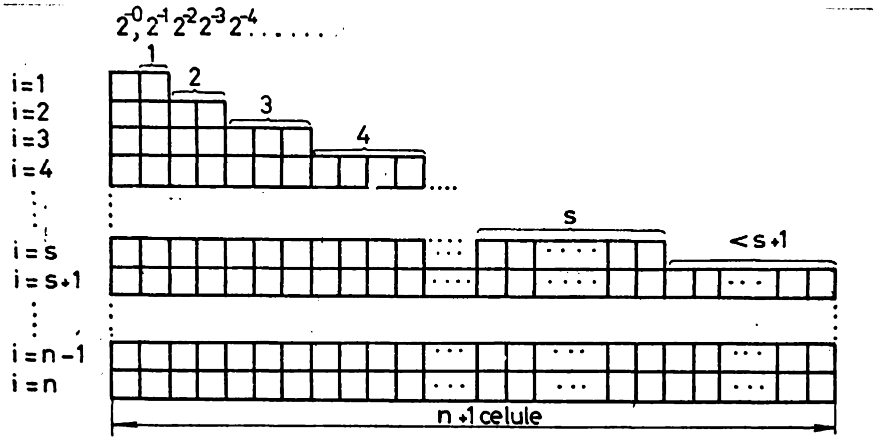


Fig.3-9. Conturul structurii de descompunere în factori

Circuitele logice ale structurii necesită la intrări nivele de tensiune. Astfel, avînd aplicat la intrări numărul $\overline{A-1}$, după un proces tranzitoriu de stabilire a funcțiilor logice ale celulelor, la ieșirile din partea stîngă ale structurii se obțin valorile corecte ale negațiilor coeficienților de creștere.

Durata operației de descompunere în factori depinde deci de timpul de propagare a nivelelor logice prin structură.

Pentru a se analiza modul în care se propagă transportul și împrumutul în rîndurile structurii s-a rezolvat descompunerea în factori pentru toate cazurile posibile ale unui număr cu n=6 biți după virgulă. Pe baza observațiilor făcute cu această ocazie s-a putut stabili care este forma numerelor pentru care operația de descompunere în factori are o durată mare. A fost efectuată apoi o cantitate mare de descompuneri în factori a unor numere cu 9 biți după virgulă urmărindu-se stabilirea cazului cel mai defavorabil din punct de vedere al lungimii totale a propagării consecutive a transportului și împrumutului în cele 9 rînduri ale structurii.

In Tabelul 3-2 se prezintă operația de descompunere în factori pentru numărul A=1,111111111 cînd, în afară de k_1, k_2, k_4, k_8, toți ceilalți coeficienți de creștere sînt nuli. Aceasta reprezintă cazul cel mai defavorabil din punct de vedere al duratei operației. In tabel s-au încercuit biții transportului și împrumutului din locul de generare și cei stabiliți în urma propagării de lungime maximă, care determină durata descompunerii în factori.

INSTITUTUL POLITEHNIC TIMIȘOARA BIBLIOTECA CENTRALĂ

BUPT

Tabelul 3-2

Poziția		Situația inițială											Situația finală										
		2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	k_i	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	k_i
1	$A-1$	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	1	1	
	P_0-1	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	$2^{-1}P_0$	0	1	0	0	0	0	0	0	0	0		0	1	0	0	0	0	0	0	0	0	
	T_1	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
1	s_1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
	I_1	1	1	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	P_1-1	0	0	0	0	0	0	0	0	0	0		0	1	0	0	0	0	0	0	0	0	
	$2^{-2}P_1$	0	0	1	0	0	0	0	0	0	0		0	0	1	1	0	0	0	0	0	0	
	T_2	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
2	s_2	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1
	I_2	1	1	1	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	P_2-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	0	0	0	0	0	0	
	$2^{-3}P_2$	0	0	0	1	0	0	0	0	0	0		0	0	0	1	1	1	1	0	0	0	
	T_3	0	0	0	0	0	0	0	0	0	0		0	1	1	1	0	0	0	0	0	0	
3	s_3	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0
	I_3	1	1	1	1	0	0	0	0	0	0		1	0	0	0	0	0	0	0	0	0	
	P_3-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	0	0	0	0	0	0	
	$2^{-4}P_3$	0	0	0	0	1	0	0	0	0	0		0	0	0	0	1	1	1	1	0	0	
	T_4	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
4	s_4	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1
	I_4	1	1	1	1	1	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	P_4-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
	$2^{-5}P_4$	0	0	0	0	0	1	0	0	0	0		0	0	0	0	0	1	1	1	1	1	
	T_5	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
5	s_5	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0
	I_5	1	1	1	1	1	1	0	0	0	0		1	0	0	0	0	0	0	0	0	0	
	P_5-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
	$2^{-6}P_5$	0	0	0	0	0	0	1	0	0	0		0	0	0	0	0	0	1	1	1	1	
	T_6	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
6	s_6	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	0
	I_6	1	1	1	1	1	1	1	0	0	0		1	0	0	0	0	0	0	0	0	0	
	P_6-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
	$2^{-7}P_6$	0	0	0	0	0	0	0	1	0	0		0	0	0	0	0	0	0	1	1	1	
	T_7	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
7	s_7	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0
	I_7	1	1	1	1	1	1	1	1	0	0		1	0	0	0	0	0	0	0	0	0	
	P_7-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	0	0	
	$2^{-8}P_7$	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	1	
	T_8	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
8	s_8	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1
	I_8	1	1	1	1	1	1	1	1	1	0		0	0	0	0	0	0	0	0	0	0	
	P_8-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	1	1	
	$2^{-9}P_8$	0	0	0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	0	0	1	
	T_9	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	1	1	
9	s_9	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
	I_9	1	1	1	1	1	1	1	1	1	1		1	0	0	0	0	0	0	0	0	0	
	P_9-1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	1	1	1	1	1	

Se observă că transportul şi împrumutul se generează şi apoi se propagă într-un număr de celule $N_{cit}= 35$.

Din exemplele analizate au rezultat următoarele concluzii:

a) Pînă în momentul inițial, cînd se aplică numărul A-1, structura logică se află "în aşteptare". Bitul 1 cel mai semnificativ al produsului deplasat fiind disponibil în permanență (fig. 3-7) în structură este stabilită o stare inițială prezentată în Tabelul 3-2. După aplicarea la intrarea structurii a numărului A-1 în structură, are loc un proces transitoriu în urma căruia se stabileşte starea finală.

b) Durata procesului va depinde de timpul de generare şi de propagare consecutivă a transportului şi împrumutului din toate rîndurile structurii. Astfel, de exemplu, împrumutul din rîndul 5, coloana 2^0, apare după ce s-a generat un transport în coloana 2^{-5} şi s-a propagat pînă în coloana 2^0 unde suma devine 1 şi generează un împrumut. Pe baza împrumutului din coloana 2^0 se stabileşte valoarea coeficientului de creştere $k_5=0$ iar acesta determină apariția la ieşirea celulelor din rîndul 5 a biților produsului corect P_5-1.

c) Dacă numărul A-1 conține numai biți 1 după virgulă, din coloana 2^{-1} pînă în coloana unde produsul deplasat conține bitul 1 cel mai semnificativ (2^{-i}), atunci transportul cu propagare de lungime maximă este generat în această ultimă coloană 2^{-i}. Situația apare numai în rîndurile unde $k_i=0$.

In concluzie, cazul cel mai defavorabil din punct de vedere al duratei de descompunere în factori este acela în care numărul A-1 are un şir de biți neîntrerupt cît mai lung egali cu 1 după virgulă iar în urma descompunerii rezultă un număr minim de coeficienți de creştere egali cu 1. Această situație corespunde numărului A-1 cu toți biții egali cu 1 după virgulă, cînd numai coeficienții de creştere din rîndurile cu număr de ordine puteri ale lui 2 sînt egali cu 1 ($k_1, k_2, k_4, k_8, \ldots$)

d) Eroarea ce apare la calculul logaritmului binar pentru m=6 şi m=9 este mai mică decît cea dată de relația (3-20). Aceasta dovedeşte faptul că relațiile deduse în capitolul II delimitează superior în mod corect eroarea absolută posibilă la calculul logaritmilor.

Din cele prezentate mai sus rezultă că durata operației de descompunere în factori este limitată superior de suma pentru aproape toate rîndurile a timpului de generare şi propagare a transportului, de formare a coeficientului de creştere şi a produsului

BUPT

corect într-un rînd. Această concluzie a fost verificată şi prin simularea structurii logice celulare de descompunere în factori, aşa cum se va vedea mai tîrziu. Concluziile deduse pentru $n=9$ au rămas valabile şi pentru o cantitate $n>9$ biţi.

Pentru un număr A cu n biţi după virgulă, numărul celulelor în care se generează sau se propagă consecutiv transportul şi împrumutul este :

$$N_{cit} = 2+3+4+\ldots+(n+1) - \left[(2^0+1)+(2^1+1)+\ldots+(2^p+1)\right] \tag{3-35}$$

unde $2^p \leq n$ iar termenii ce se scad represintă numărul de celule din rîndurile corespunzătoare puterilor lui 2 (unde $k_i=1$) în care nu apare generare şi propagare de transfer sau împrumut. Relaţia (3-35) se mai poate pune în forma :

$$N_{cit} = \frac{n(n+3)}{2} - (2^{p+1}+p) \tag{3-36}$$

care pentru $n=28$ conduce la $N_{cit}= 398$.

Se notează cu t_1 timpul de generare sau propagare a transportului sau împrumutului peste o celulă, cu t_2 - timpul de propagare al variabilei logice $\bar{k}_i$ peste două circuite NU-SI de putere ce comandă barele k_i şi $\bar{k}_i$ ale celulelor dintr-un rînd, cu t_3 - timpul de formare a produsului corect $P_{i,j}$ cînd este disponibilă valoarea bitului $\bar{k}_i$ şi cu t_4 - timpul de formare a bitului sumei $s_{i,j}$ după ce încetează propagarea transportului. Cu aceasta, durata operaţiei de descompunere în factori pentru cazul cel mai defavorabil este :

$$t_{dfmax} = N_{cit}\cdot t_1+(n_0+1)t_2+n(t_3+t_4), \tag{3-37}$$

unde $n_0= n-p-1$ este numărul de rînduri cu $k_i=0$.

Dacă celula din fig.3-8 se realizează cu circuite integrate SI-NU şi NU pentru sumă, SI-SAU pentru transport, împrumut şi produs, SI-NU de putere pentru coeficientul de creştere,de tipul celor din seria rapidă [69] atunci :

$$t_1 = t_p \ ; \ t_2 = 2t_p \ ; \ t_3 = t_p \ ; \ t_4 = 2t_p$$

t_p fiind timpul de propagare tipic pe un circuit integrat.

Cu aceasta durata maximă a operaţiei de descompunere în factori, pentru cazul cînd A-1 se aplică la intrare cu n biţi este:

$$t_{dfmax_n} = N_{cit}\cdot t_p+ 2(n-p)t_p+ 3nt_p = \left[\frac{n(n+13)}{2} - 2^{p+1}-3p\right] t_p \tag{3-38}$$

Pentru $n=28$, $p=4$, $n_0=23$ se obţine astfel un timp $t_{dfmax}= 530t_p$ iar pentru $t_p= 6$ ns rezultă :

$$t_{dfmax_n} = 3180 \text{ ns} = 3,18 \ \mu s \tag{3-39}$$

BUPT

Dacă numărul A-1 se aplică la intrarea structurii logice celulare de descompunere în factori cu numai m biți ($n = m+4...6$) atunci cazul cel mai defavorabil care stabilește timpul t_{dfmax} apare pentru A-1 = 0,01100...00. In acest caz

$$N_{cit} = \frac{(n+2)(n-1)}{2} - (2^{p+1}+p+2) \qquad (3\text{-}40)$$

și $k_2, k_4, k_5, k_8, k_{16}, k_{28}...$ sînt egali cu 1.

Pentru $n=28$, $m=23$, $n_o=22$, $N_{cit}=367$ și $t_p = 6$ ns rezultă :

$$\underline{t_{dfmax_m} = 497\ t_p = 2{,}982\ \mu s} \qquad (3\text{-}41)$$

Inafara unui număr redus de cazuri, cum sînt și cele prezentate mai sus, descompunerea în factori durează un timp mult mai scurt deoarece transportul și împrumutul nu se propagă peste un număr la fel de mare de celule.

Durata minimă a operației de descompunere în factori apare pentru situația :

$$A = 1{,}00...00$$

cînd în structura logică nu are loc nici o modificare a stării inițiale. In acest caz durata minimă a operației de descompunere în factori este :

$$\underline{t_{dfmin} = 0}$$

In concluzie durata operației de descompunere în factori este cuprinsă între 0 și t_{dfmax} în funcție de numărul ce se descompune.

Structura logică celulară de însumare a termenilor.

In această structură se face însumarea termenilor $L_1...L_n$ în funcție de valorile coeficienților de creștere $k_1...k_n$ determinați în structura logică celulară de descompunere în factori.

Modul de realizare a acestei structuri este prezentat în fig.3-10. Celulele acestei structuri conțin circuitele logice din fig.3-11.

Intr-un rînd i al structurii logice de însumare a termenilor (fig.3-10) se adaugă la suma efectuată anterior un nou termen k_iL_i care poate fi egal cu L_i (cînd $k_i=1$) sau cu 0 (cînd $k_i=0$). Operația se desfășoară în modul următor :

- se adună suma anterioară :

$$S_{i-1} = \sum_{i=1}^{i=i-1} k_iL_i$$

cu termenul L_i corespunzător rîndului i și rezultă :

$$s_i' = \sum_{i=1}^{i=i-1} k_iL_i + L_i \qquad (3\text{-}42)$$

BUPT

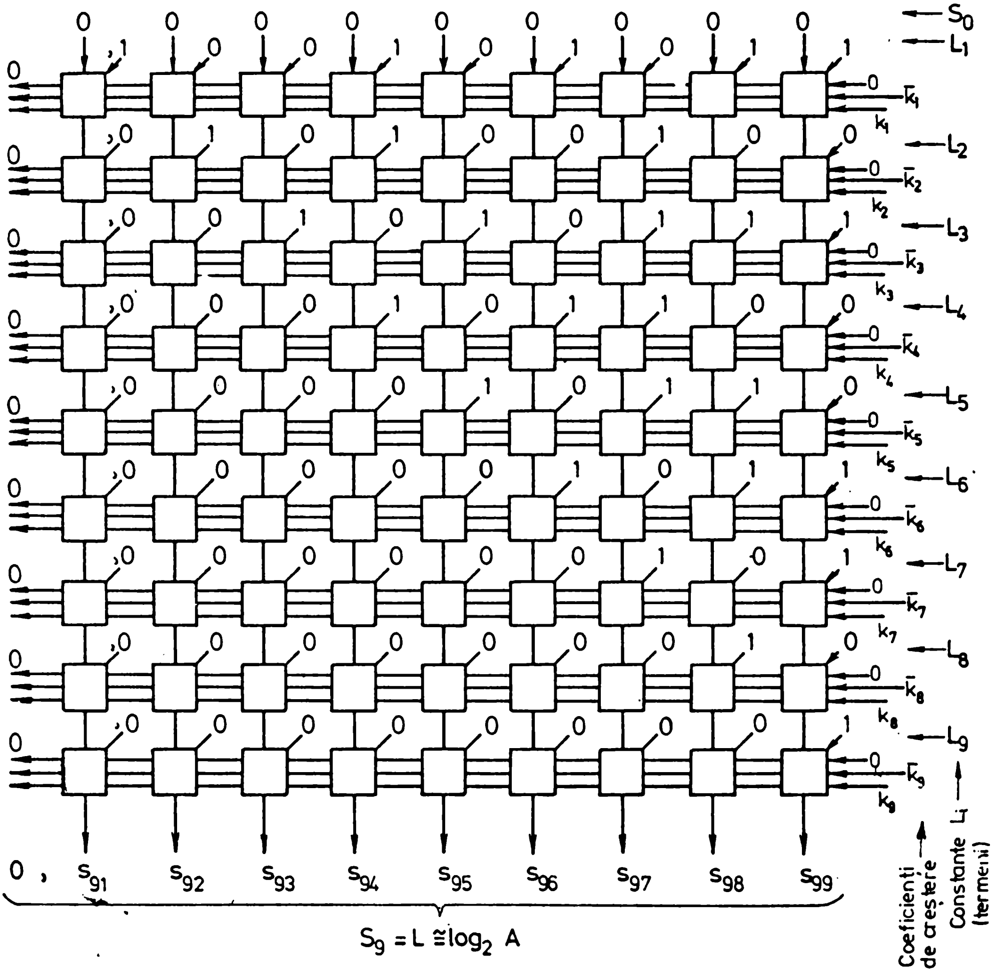


Fig.3-lo. Structură logică celulară de însumare a termenilor.

- se transmite spre rîndul următor suma nou obținută s_i' dacă $k_i=1$ sau suma anterioară S_{i-1} nemodificată dacă $k_i=0$. Astfel, la ieșirea rîndului i se obține de fapt suma :

$$S_i = \sum_{i=1}^{i=i} k_i L_i$$

Transmiterea sumei de la un rînd la altul se face pe verticală în fig.3-lo. Termenii L_i se aplică la intrările diagonale ale celulelor.

Ca sumă inițială se ia S_0=0,000...000. Deoarece rezultatul L = $\log_2 A$ va fi cuprins în domeniul :

$$0 \leq \log_2 A < 1$$

el va avea întotdeauna bitul întreg egal cu zero, nu apare la însumare un transport în prima coloană din dreapta virgulei și

BUPT

deci în stînga virgulei nu mai este necesară o coloană (în această coloană nu se efectuează operaţii aritmetice). Astfel structura generează numai mantisa logaritmului.

In cadrul celulei din rîndul i, coloana j, utilizată în structura logică celulară de însumare a termenilor (fig.3-11) se realizează următoarele funcţii logice :

$$s'_{i,j} = S_{i-1,j} \oplus L_{i,j} \oplus T'_{i,j+1} \tag{3-43}$$

$$T'_{i,j} = S_{i-1,j}L_{i,j}+S_{i-1,j}T'_{i,j+1}+L_{i,j}T'_{i,j+1} \tag{3-44}$$

$$S_{i,j} = S_{i-1,j}\bar{k}_i+s'_{i,j}k_i \tag{3-45}$$

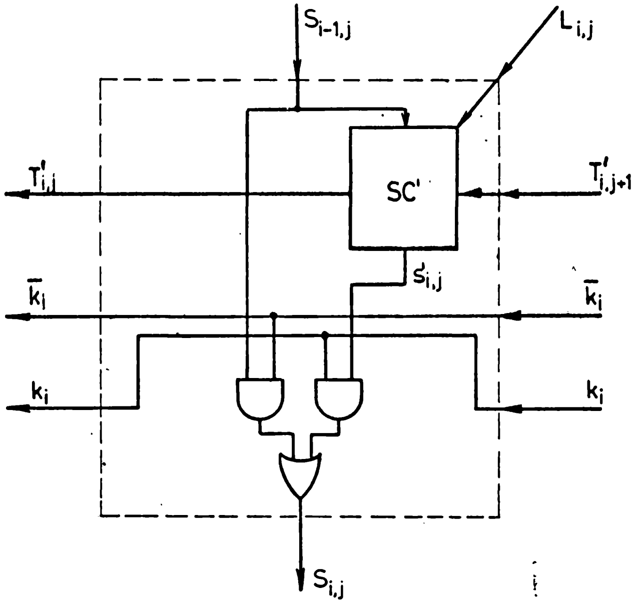


Fig.3-11. Celula structurii de însumare a termenilor.

Funcţia logică $s'_{i,j}$ reprezintă suma logică a biţilor $S_{i-1,j}$; $L_{i,j}$; $T'_{i,j+1}$ iar funcţia $T'_{i,j}$ reprezintă transportul rezultat la adunarea acestora în sumatorul complet SC'. Funcţia $S_{i,j}$ reprezintă bitul sumei corecte, fiind egal cu S_{i-j} cînd $k_i=0$ sau cu $s'_{i,j}$ cînd $k_i=1$.

Structura logică celulară din fig.3-10 conţine un număr de celule

$$N'_c = n^2 \tag{3-46}$$

mai mare decît cel necesar din cauza însumării termenilor începînd cu ordinul i=1. Deasemenea, intrările sumatoarelor complete SC' din celule nu sînt folosite eficient-la un număr mare din intrările diagonale se aplică bitul 0 - deci ele nu sînt utilizate. In acele celule se pot utiliza doar semisumatoare sau este posibilă o re-

BUPT

structurare a circuitelor astfel încît să se utilizeze numai celulele la care $L_{i,j}=1$. Totuşi, avînd în vedere posibilitatea comasării structurilor logice celulare de descompunere în factori, de însumare a termenilor şi a structurilor utilizate la generarea antilogaritmilor, este necesar ca structura logică de însumare a termenilor să rămînă în această formă generală (fig.3-1o).

Pentru aprecierea duratei de însumare a termenilor şi a felului în care ea influenţează asupra duratei operaţiei complete de generare a logaritmului, este necesar să se analizeze modul cum se desfăşoară însumarea termenilor în timp. Inainte de determinarea valorii corecte a coeficientului de creştere într-un rînd i al structurii logice de descompunere în factori (prin stabilirea împrumutului în celula din poziţia 2^0) are loc adunarea termenului L_i la suma anterioară S_{i-1} obţinîndu-se suma s_i' . Această operaţie se efectuează în paralel cu cea de descompunere în factori. In acest timp împrumutul din celula din poziţia 2^0 a structurii de descompunere este egal cu 0 deci $\bar{k}_i=0$ şi $k_i=1$ astfel încît se stabileşte şi o sumă provizorie S_i. In concluzie înainte de încheierea descompunerii în factori în structura logică de însumare a termenilor se adună în fiecare rînd termenul corespunzător. Aceasta este starea iniţială a structurii.

După determinarea valorii corecte a unui coeficient de creştere k_i, prin funcţia logică (3-45) se stabileşte numărul ce se transmite spre rîndul următor al structurii de însumare a termenilor : s_i' dacă $k_i=1$ sau S_{i-1} dacă $k_i=0$. Din momentul în care se determină primul coeficient de creştere $k_i=0$ suma S_i şi sumele din toate rîndurile următoare se modifică faţă de starea iniţială. In structura logică de însumare a termenilor intervine un regim tranzitoriu a cărui durată depinde de numărul de coeficienţi de creştere egali cu 0 şi de momentul determinării acestora.

Operaţia de însumare a termenilor are deci loc în paralel cu cea de descompunere în factori. Din exemplele cercetate rezultă că propagarea transportului în rîndurile structurii de însumare a termenilor are o lungime medie redusă (constantele ce se adună au un număr redus de biţi 1).

Situaţia cea mai defavorabilă din punct de vedere al duratei operaţiei de însumare a termenilor apare cînd în ultimul rînd al structurii are loc generarea şi propagarea transportului peste n-2 celule. Situaţia corespunde logaritmului L = 0,1100... (pentru n=28 $k_1, k_4 \dots k_6, k_8 \dots k_{11}, k_{13} \dots k_{18}, k_{20}, k_{21}, k_{27}, k_{28}$ sînt egali cu 1). Acest transport lung în ultimul rînd al structurii logice

BUPT

de însumare a termenilor este provocat de însumarea constantei $L_n=2^{-n}$ cînd $k_n=1$. Dacă timpul de propagare al circuitului SI-SAU ce realizează transportul este t_p atunci durata propagării transportului peste n-2 celule este $(n-2)t_p$.

Situaţia de mai sus nu coincide însă şi cu o propagare de lungime maximă a transportului în rîndul n al structurii logice celulare de descompunere în factori deoarece $k_n=1$. Acest lucru este important deoarece arată că operaţia de însumare a termenilor nu poate prelungi operaţia de generare a logaritmului peste timpul de descompunere în factori maxim t_{dfmax}, calculat anterior şi care corespunde situaţiei $k_n=0$.

Prin urmare durata totală a operaţiei de generare a logaritmului binar al unui număr este :

$$t_{log} \leq t_{dfmax} \qquad (3\text{-}47)$$

Din acest motiv în cadrul cercetărilor s-a acordat o atenţie deosebită operaţiei de descompunere în factori şi în special propagării transportului şi împrumutului urmărindu-se realizarea unei viteze mărite.

3.3.2. Structuri logice celulare pentru generarea antilogaritmilor binari /37/.

Calculul antilogaritmului binar al unui număr L, prezentat în paragraful 2.2, include următoarele operaţii aritmetice :

- descompunerea numărului L în termeni de forma

$$L_i = \log_2(1+2^{-i})$$

(adică determinarea coeficienţilor de creştere k_i),

- înmulţirea antilogaritmilor termenilor, adică înmulţirea factorilor $1+2^{-i}$ corespunzători valorilor $k_i = 1$.

Aceste două operaţii se pot desfăşura în paralel cu ajutorul circuitului din fig.3-12.

In fig.3-13 şi 3-14 se prezintă structurile logice celulare de descompunere în termeni respectiv de înmulţire a factorilor pentru n=9 biţi. Antilogaritmul A se obţine cu m biţi pentru a fi utilizaţi în calcule în calculator. In fig.3-13 şi 3-14 s-au folosit m=6 biţi. Fiecare din cele două structuri logice este compusă din celule identice avînd schema logică din fig.3-15 respectiv 3-16.

Structura logică celulară de descompunere în termeni.

Intr-un rînd i al structurii logice celulare din descompunere în termeni se efectuează următoarele operaţii :

BUPT

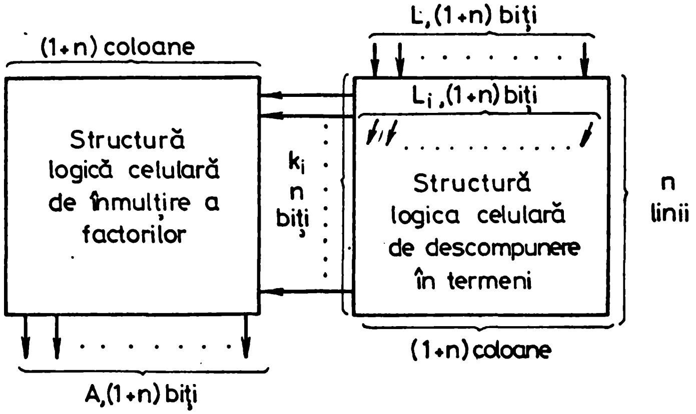


Fig.3-12. Structurile logice celulare de generare a logaritmilor binari.

- adunarea la suma anterioară

$$S_{i-1} = \sum_{i=1}^{i=i-1} k_i L_i \qquad (3-48)$$

(suma termenilor incluşi în dezvoltare pînă la rîndul i-1) a termenului L_i corespunzător rîndului i. Adică se face operaţia :

$$s'_i = S_{i-1} + L_i \qquad (3-49)$$

- compararea sumei nou obţinută s'_i cu numărul L ce se descompune în termeni (compararea se face prin determinarea cifrei împrumutului la scădere în fiecare celulă şi propagarea lui spre celula din stînga),

- transmiterea spre rîndul următor (i+1) a sumei de la intrare, S_{i-1} nemodificate cînd k_i=0 ($\bar{k}_i$=1) sau a sumei noi s'_i cînd k_i= 1 ($\bar{k}_i$= 0).

Ca sumă iniţială se ia S_0 = 0,00..00 iar dacă numărul L ce se antilogaritmează conţine numai m biţi el se completează cu zerouri pînă la n biţi. In structură se utilizează numai negaţia biţilor lui L.

In cazul unei descompuneri exacte biţii numărului L şi ai sumei S_n de la ieşirea structurii trebuie să coincidă.

Coloana din stînga virgulei este necesară deoarece suma s'_i poate deveni egală sau mai mare decît 1 (cel mult 1,25).

Ieşirea de împrumut de la celula din stînga a fiecărui rînd ($\bar{k}_i$) şi negaţia ei (k_i) se retrimit prin toate celulele rîndului respectiv pentru a controla transmiterea spre rîndul următor a sumei corecte a termenilor.

BUPT

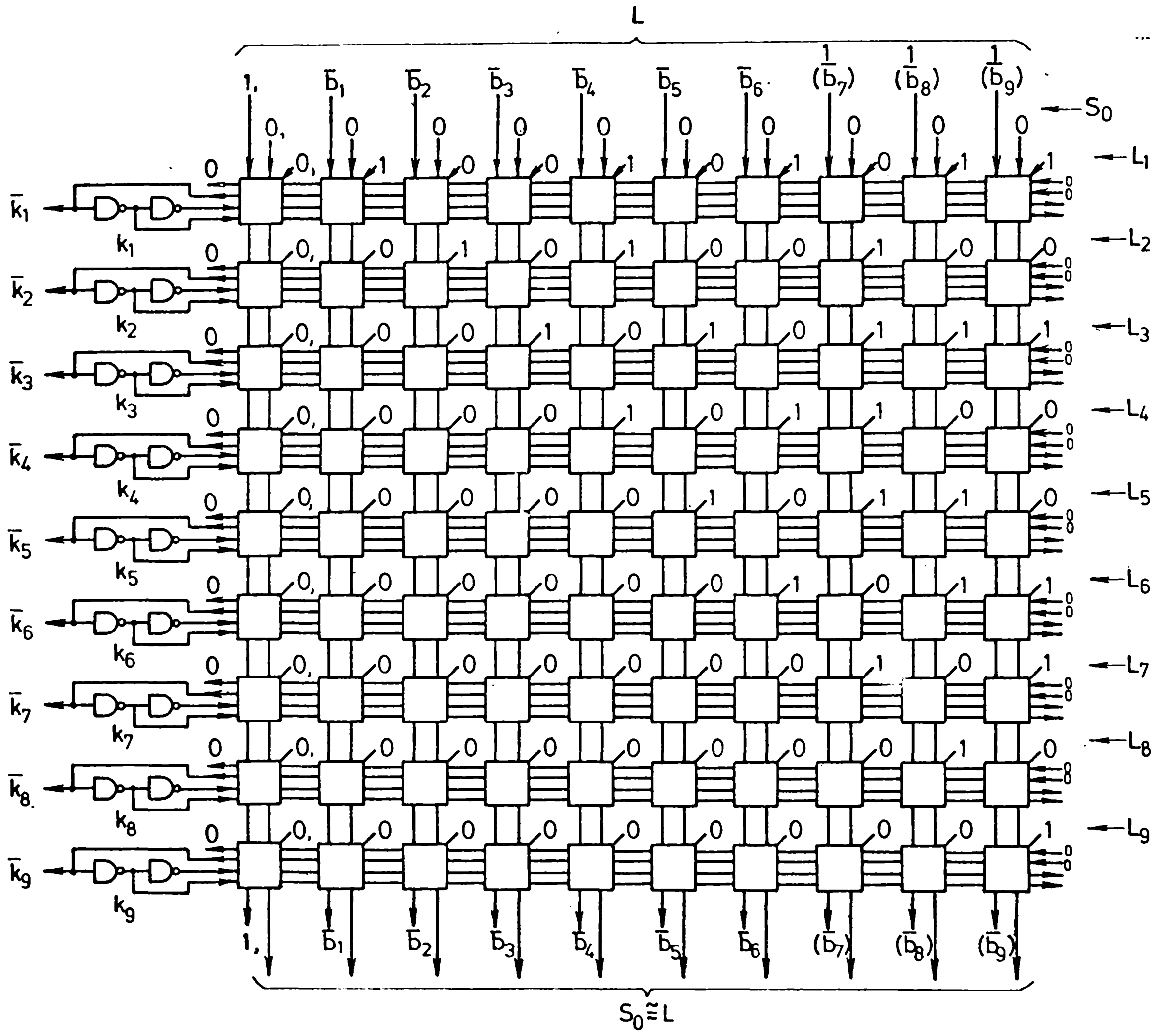


Fig.3-13 Structură logică celulară de descompunere în termeni.

INSTITUTUL POLITEHNIC TIMIȘOARA BIBLIOTECA CENTRALĂ

BUPT

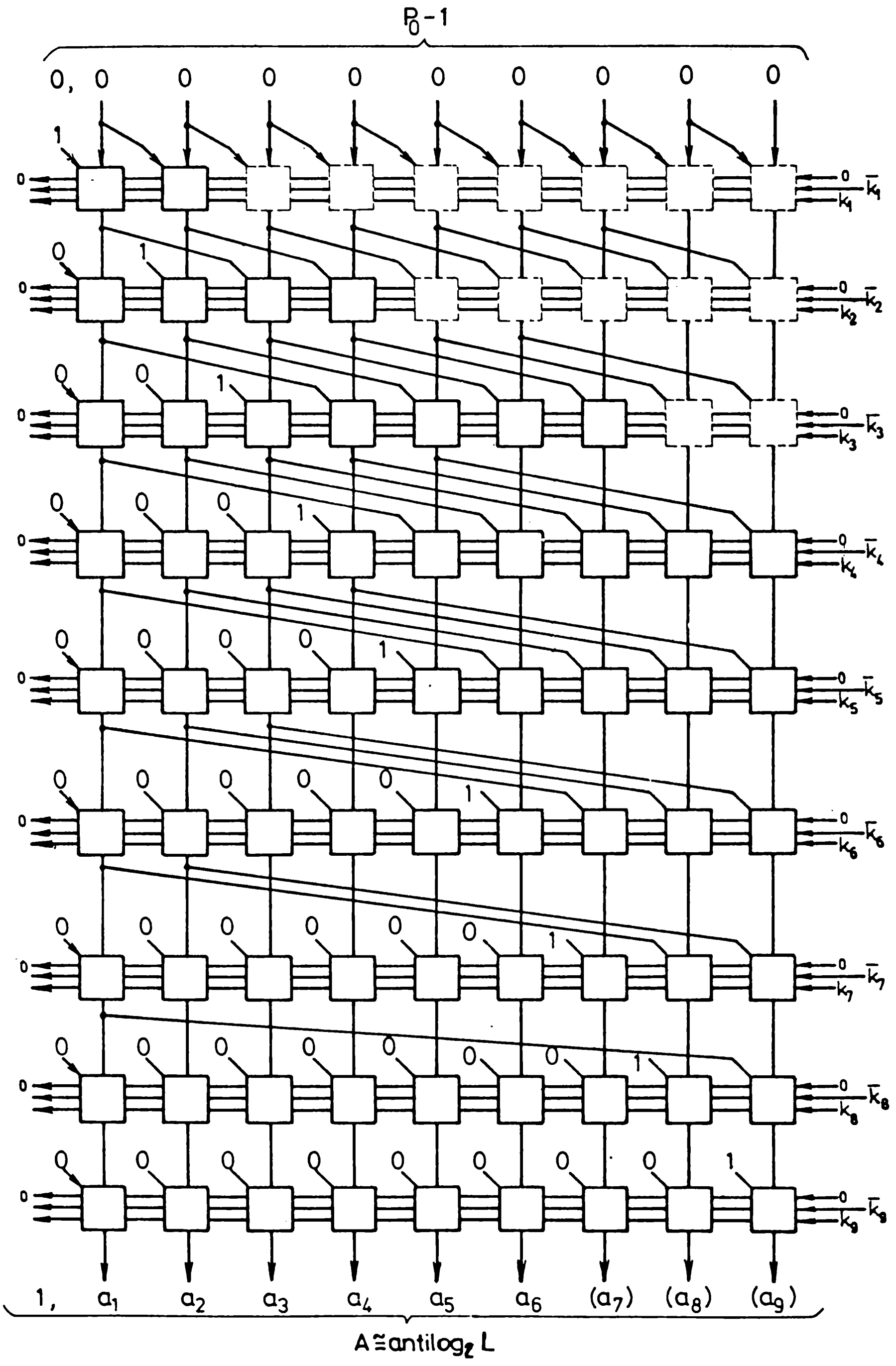


Fig.3-14. Structură logică celulară de înmulţire a factorilor.

BUPT

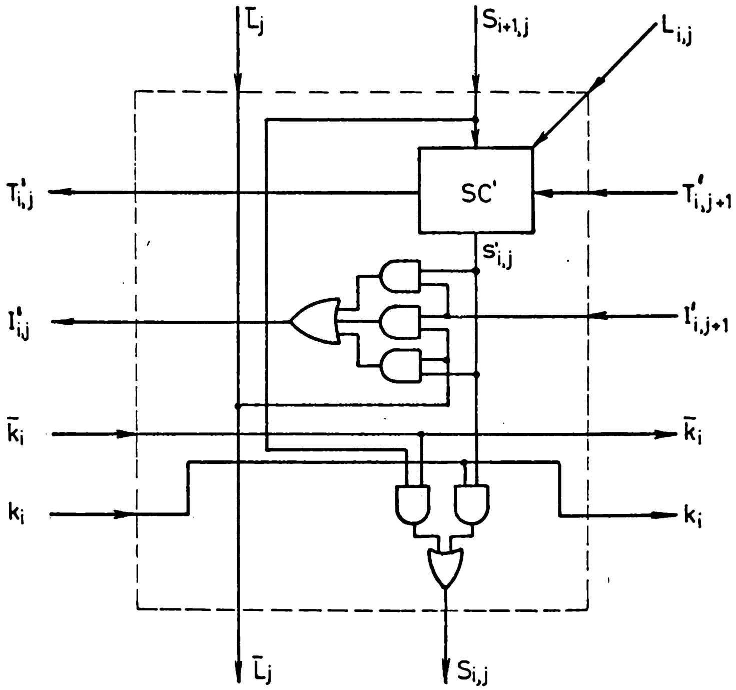
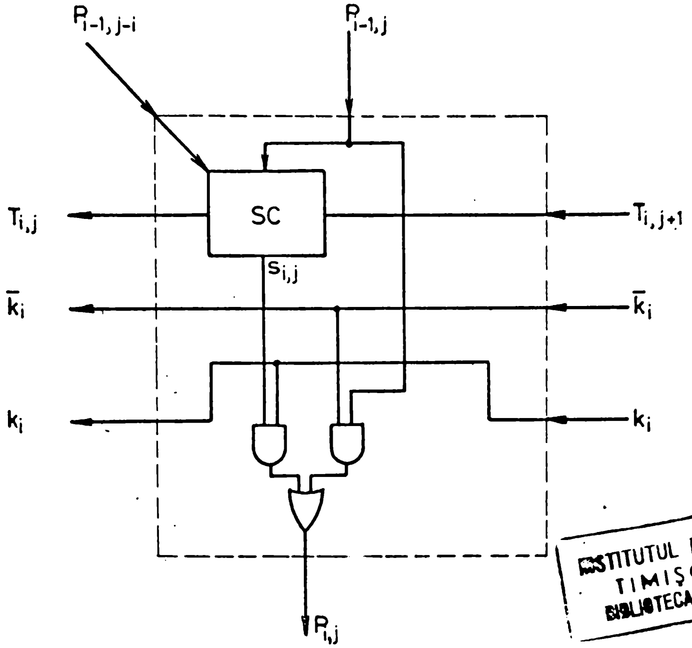


Fig.3-15 Celula structurii de descompunere în termeni.



INSTITUTUL POLITEHNIC
TIMIŞOARA
BIBLIOTECA CENTRALĂ

Fig.3-16. Celula structurii de înmulţire a factorilor.

BUPT

In cadrul celulei din rîndul i, coloana j (fig.3-13) se realizează următoarele funcţii logice :

$$s'_{i,j} = S_{i-1,j} \oplus L_{i,j} \oplus T'_{i,j+1} \qquad (3\text{-}50)$$

$$T'_{i,j} = S_{i-1,j}L_{i,j} + S_{i-1,j}T'_{i,j+1} + L_{i,j}T'_{i,j+1} \qquad (3\text{-}51)$$

$$I_{i,j} = \bar{L}_j(s'_{i,j} + I'_{i,j+1}) + s'_{i,j}\bar{I}'_{i,j+1} \qquad (3\text{-}52)$$

$$S_{i,j} = S_{i-1,j}\bar{k}_i + s'_{i,j}k_i \qquad (3\text{-}53)$$

care reprezintă suma provizorie, transportul, împrumutul şi suma corectă.

Se observă o mare asemănare între operaţia de descompunere în factori de la calculul logaritmului şi operaţia de descompunere în termeni de la calculul antilogaritmului. De aceea rezultă şi celule cu conţinut identic (fig.3-8 şi 3-15). Singura deosebire în ceea ce priveşte variabilele de intrare este că la intrările diagonale se aplică variabile diferite : $P_{i-1,j-i}$ respectiv $L_{i,j}$.

Structura logică celulară de descompunere în termeni necesită un număr de n(1+n) celule identice adică acelaşi număr de celule ca şi structura logică celulară de descompunere în factori dacă nu se elimină celulele din colţul din dreapta, sus (fig.3-7). Cele două structuri logice diferă astfel numai prin conexiunile de la intrările diagonale ale celulelor.

Funcţionarea structurii logice celulare de descompunere în termeni va fi asemănătoare cu cea a structurii de descompunere în factori descrisă anterior. Rămîn deci valabile şi aici o serie de considerente privind durata operaţiei de descompunere în termeni.

Durata maximă a operaţiei de descompunere în termeni se poate calcula cu aceeaşi relaţie (3-37) în care însă numărul de celule N_{cit} peste care se propagă transportul şi împrumutul este altul.

Durata maximă a operaţiei de descompunere în termeni apare în cazul cînd numărul ce se descompune conţine o cantitate cît mai mare de biţi 1 în dreapta virgulei şi o cantitate cît mai mică de coeficienţi de creştere egali cu 1.

Pentru cazul în care numărul L se aplică la intrarea structurii cu n biţi (cu $n \leq 30$) această situaţie apare dacă $L = L_4$ adică numai coeficientul $k_4 = 1$ (Tabelul 3-3).

Pentru cazul în care numărul L se aplică la intrare cu m biţi (m<n), corespunzători domeniului în care erorile sînt neglijabile la generarea antilogaritmului, atunci situaţia cea mai defavorabilă apare cînd $L = L_5 + L_{26}$ şi deci numai k_5 şi k_{26} sînt egali cu 1.

BUPT

Tabelul 3-3

Poziția		Situația inițială											Situația finală										
		2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	k_1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	k_1
1	L	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	
	S_0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
1	L_1	0	1	0	0	1	0	1	0	1	1		0	1	0	0	1	0	1	0	1	1	
	T_1'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	s_1'	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0	1	0	1	0	1	1	0
	I_1'	1	1	1	1	1	1	1	1	1	1		1	1	0	0	0	0	0	0	1	1	
	S_1	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
2	L_2	0	0	1	0	1	0	0	1	0	0		0	0	1	0	1	0	0	1	0	0	
	T_2'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	s_2'	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0
	I_2'	1	1	1	1	1	1	1	1	0	0		1	1	1	0	0	0	0	0	0	0	
	S_2	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
3	L_3	0	0	0	1	0	1	0	1	1	1		0	0	0	1	0	1	0	1	1	1	
	T_3'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	s_3'	0	0	0	1	0	1	0	1	1	1	0	0	0	0	1	0	1	0	1	1	1	0
	I_3'	1	1	1	1	1	1	1	1	1	1		1	1	1	1	0	1	0	1	1	1	
	S_3	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
4	L_4	0	0	0	0	1	0	1	1	0	0		0	0	0	0	1	0	1	1	0	0	
	T_4'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	s_4'	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	1	1	0	0	1
	I_4'	1	1	1	1	1	1	1	1	0	0		0	0	0	0	0	0	0	0	0	0	
	S_4	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	
5	L_5	0	0	0	0	0	1	0	1	1	0		0	0	0	0	0	1	0	1	1	0	
	T_5'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1	1	1	0	0	
	s_5'	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0
	I_5'	1	1	1	1	1	1	1	1	1	0		1	1	1	1	0	0	0	0	1	0	
	S_5	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	
6	L_6	0	0	0	0	0	0	1	0	1	1		0	0	0	0	0	0	1	0	1	1	
	T_6'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0	0	0	
	s_6'	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0	1	1	1	0
	I_6'	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	0	1	1	1	
	S_6	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	
7	L_7	0	0	0	0	0	0	0	1	0	1		0	0	0	0	0	0	0	1	0	1	
	T_7'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	1	0	0	
	s_7'	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	1	0
	I_7'	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	0	0	1	1	
	S_7	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	
8	L_8	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0	
	T_8'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	s_8'	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	0	0
	I_8'	1	1	1	1	1	1	1	1	1	0		1	1	1	1	1	1	1	1	1	0	
	S_8	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	
9	L_9	0	0	0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	0	0	1	
	T_9'	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	
	s_9'	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	0	1	0
	I_9'	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	
	S_9	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	0	1	1	0	0	

In ambele cazuri transportul se generează în apropierea poziţiei 2^{-1} din fiecare rînd în care $k_i=0$, începînd cu rîndul corespunzător primului coeficient $k_i=1$ şi se propagă spre stînga. Intr-o poziţie oarecare transportul cauzează generarea unui împrumut care se propagă în continuare spre stînga pînă în coloana 2^0 unde determină valoarea coeficientului de creştere. Lungimea maximă a propagării transportului şi împrumutului, N_{cit}, nu se poate aici calcula exact analitic ca în cazul descompunerii în factori. Ea trebuie determinată pentru un n dat desfăşurînd operaţiile pe hîrtie. Cu aproximaţie se poate evalua N_{cit} cu relaţia :

$$N'_{cit} \approx \frac{(n+7)(n-4)}{2} \qquad (3-55)$$

pentru cazul cînd numărul L vine la intrarea structurii cu n biţi sau :

$$N'_{cit} \approx \frac{(n+7)(n-5)}{2} \qquad (3-56)$$

în cazul cînd numărul L vine cu m biţi (n=m+4...6).

Pentru calculul timpului se poate utiliza relaţia (3-57) în care numărul n_o este uşor de stabilit deoarece fie numai $k_4=1$ fie numai k_5 şi k_{26} ($n_o=n-1$ respectiv $n-2$).

Dacă n=28 şi numărul L se aplică la intrare cu n biţi atunci $n_o=n-1$ şi $N'_{cit}=417$ (determinat exact). Pentru acelaşi tip de circuite ca şi mai înainte rezultă :

$$t_{dtmax} = N'_{cit}t_p+(n_o+1)2\,t_p+ n\,3\,t_p \qquad (3-57)$$

sau :

$$\underline{t_{dtmax_n} = 557\,t_p = 3{,}342\ \mu s} \qquad (3-58)$$

Dacă n=28 şi numărul L se aplică la intrare cu m=23 biţi atunci pentru cazul cel mai defavorabil $N'_{cit}= 403$, $n_o= n-2$ şi

$$\underline{t_{dtmax_m} = 541\,t_p= 3{,}246\ \mu s} \qquad (3-59)$$

Durata minimă a operaţiei de descompunere în termeni se obţine în cazul cînd în structură nu au loc modificări ale stării iniţiale deci cînd L=0,00...00, situaţie asemănătoare cu cea de la structura de descompunere în factori. Rezultă deci $t_{dtmin}= 0$ şi prin urmare descompunerea în termeni se realizează într-un timp cuprins între 0 şi t_{dtmax}, în funcţie de valoarea numărului ce se descompune.

Structura logică celulară de înmulţire a factorilor.

In această structură logică (fig.3-14) se face înmulţirea factorilor de forma $(1+2^{-i})$ în funcţie de valorile coeficienţilor de creştere $k_1...k_n$ determinate în structura de descompunere în

BUPT

termeni.

Intr-un rînd i al structurii se efectuează următoarele operaţii aritmetice şi logice :

- înmulţirea produsului anterior :

$$P_{i-1} = \prod_{i=1}^{i=i-1} (1+k_i 2^{-i})$$

(produsul factorilor corespunzători termenilor incluşi în desvoltare pînă în rîndul i-1) cu factorul corespunzător rîndului i rezultînd :

$$s_i = P_{i-1}(1+2^{-i})$$

După cum s-a arătat înmulţirea se reduce la o deplasare şi o adunare ;

- se transmite spre rîndul următor produsul nou obţinut -s_i- dacă k_i=1 sau a produsului anterior P_{i-1} nemodificat dacă k_i=0. Astfel, la ieşirea din rîndul i se obţine de fapt produsul:

$$P_i = \prod_{i=1}^{i=i} (1+k_i 2^{-i}) \tag{3-60}$$

Si aici, pentru eliminarea unei coloane din stînga virgulei, se operează numai cu partea fracţionară a produsului: P_i-1. Deoarece nu apare niciodată un transport în coloana din dreapta virgulei ($1 \leqslant P_i < 2$), în coloana din stînga virgulei nu se efectuează operaţii aritmetice. Se determină astfel cu ajutorul structurii numărul A-1, adică partea fracţionară a lui A (A este antilogaritmul lui L).

Transmiterea produsului spre un rînd următor se face pe verticală iar deplasarea lui este realizată prin conexiunile diagonale. Ca produs iniţial se ia P_0= 1,oo...oo (adică P_0-1=o,oo...oo).

La ieşire se poate ataşa părţii fracţionare a antilogaritmului întregul 1 astfel ca antilogaritmul să fie cuprins în domeniul $1 \leqslant A < 2$.

In celula din rîndul i şi coloana j a structurii logice celulare de înmulţire a factorilor (fig.3-16) se realizează următoarele funcţii logice :

$$s_{i,j} = P_{i-1,j} \oplus P_{i-1,j-i} \oplus T_{i,j+1} \tag{3-61}$$

$$T_{i,j} = P_{i-1,j}\, P_{i-1,j-i} + P_{i-1,j}\, T_{i,j+1} + P_{i-1,j-i}\, T_{i,j+1} \tag{3-62}$$

$$P_{i,j} = P_{i-1,j}\, \overline{k}_i + s_{i,j} k_i \tag{3-63}$$

care reprezintă suma, transportul şi produsul corect.

BUPT

Din cauza motivelor arătate în structura logică celulară de descompunere în factori şi la structura logică de înmulţire a factorilor celulele din colţul dreapta sus (cu linie întreruptă în fig.3-14) pot să lipsească.

Sînt valabile şi aici considerentele de la structura logică celulară de însumare a termenilor privind durata operaţiei de înmulţire a factorilor. Deasemenea, problema duratei totale a operaţiei de antilogaritmare se pune în acelaşi mod. Rezultă şi aici concluzia că durata maximă a operaţiei de antilogaritmare este egală cu durata maximă a operaţiei de descompunere în termeni:

$$t_{antilog} \leq t_{dtmax} \qquad (3-64)$$

Pentru reducerea duratei operaţiei este necesar deci să se reducă timpul de propagare a transportului şi împrumutului.

3.3.3. Posibilităţi de comasare a structurilor logice celulare de generare a logaritmilor şi antilogaritmilor.

Din cele prezentate anterior se observă o mare asemănare între diferite structuri logice atît în ceea ce priveşte componenţa celulelor cît şi în privinţa interconexiunilor.

Astfel, celula structurii logice de descompunere în factori (fig.3-8) este asemănătoare cu celula structurii de descompunere în termeni (fig.3-15). Deosebirea dintre ele constă în aceea că, în locul variabilelor $\overline{A}_j$ şi $P_{i-1,j-i}$ de la prima celulă, la a doua celulă apar variabilele $\overline{L}_j$ respectiv $L_{i,j}$. Dar bara pe care se transmite $\overline{A}_j$ la generarea logaritmului se poate folosi şi pentru transmiterea lui $\overline{L}_i$ la generarea antilogaritmului, deci rămîne o singură deosebire privind intrările $P_{i-1,j-i}$ şi $L_{i,j}$.

Celula structurii logice celulare de însumare a termenilor (fig.3-11) este asemănătoare cu celula structurii de înmulţire a factorilor (fig.3-16). Deosebirea dintre ele constă în aceea că, în locul variabilei $L_{i,j}$ de la prima celulă, la a doua celulă apare variabila $P_{i-1,j-i}$.

Comparînd ansamblul celor două celule utilizate la generarea logaritmului (fig.3-8 şi 3-11) cu cel al celulelor utilizate la generarea antilogaritmului (fig.3-15 şi 3-16) se observă că cele două perechi de celule conţin aceleaşi circuite logice şi aceleaşi intrări, fiind necesar însă ca valoarea coeficienţilor de creştere să se stabilească la generarea logaritmului prin comparaţia între numărul A ce se logaritmează şi suma $s_{i,j}$ iar la generarea antilogaritmului prin comparaţia între numărul L ce se antilogaritmează şi suma $s'_{i,j}$.

Astfel cu unele completări înafara structurii logice şi o modificare simplă a celulei, o singură pereche de celule din cele de mai sus se poate utiliza atît la generarea logaritmului cît şi a antilogaritmului.

Celula comună este prezentată în fig.3-17, în ea s-au reprezentat circuitele logice ce realizează o anumită funcţie logică distinctă prin blocuri.

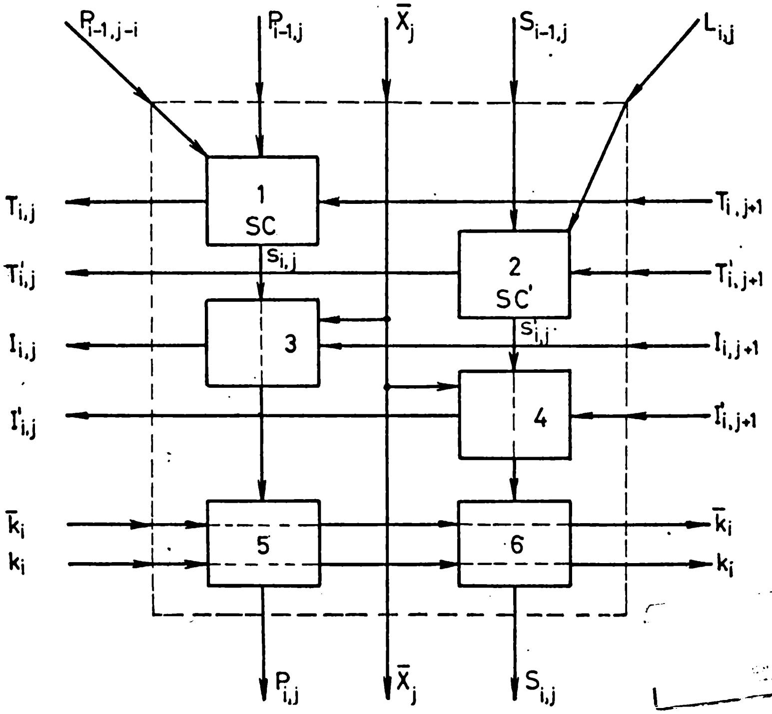


Fig.3-17. Celula structurii comasate de generare a logaritmilor şi antilogaritmilor binari.

Blocurile 1 şi 2 reprezintă sumatoare complete identice, blocurile 3 şi 4 reprezintă circuite identice ce realizează împrumutul iar blocurile 5 şi 6 stabilesc valoarea corectă a produsului sau sumei ce trebuie trimisă spre rîndul următor al structurii.

Funcţiile logice realizate de blocuri au forma cunoscută anterior (relaţiile 3-29...3-32, 3-50...3-53) în care variabilele $\bar{A}_j$ şi $\bar{L}_j$ se înlocuiesc cu $\bar{X}_j$. La generarea logaritmului la intrările $\bar{X}_j$ se aplică biţii numărului $\bar{A}$ iar la generarea antilogaritmului - biţii numărului $\bar{E}$. Logaritmul şi antilogaritmul se obţin la ieşiri diferite ale celulelor din ultimul rînd al structurii logice celulare : logaritmul se obţine la ieşirile de la

BUPT

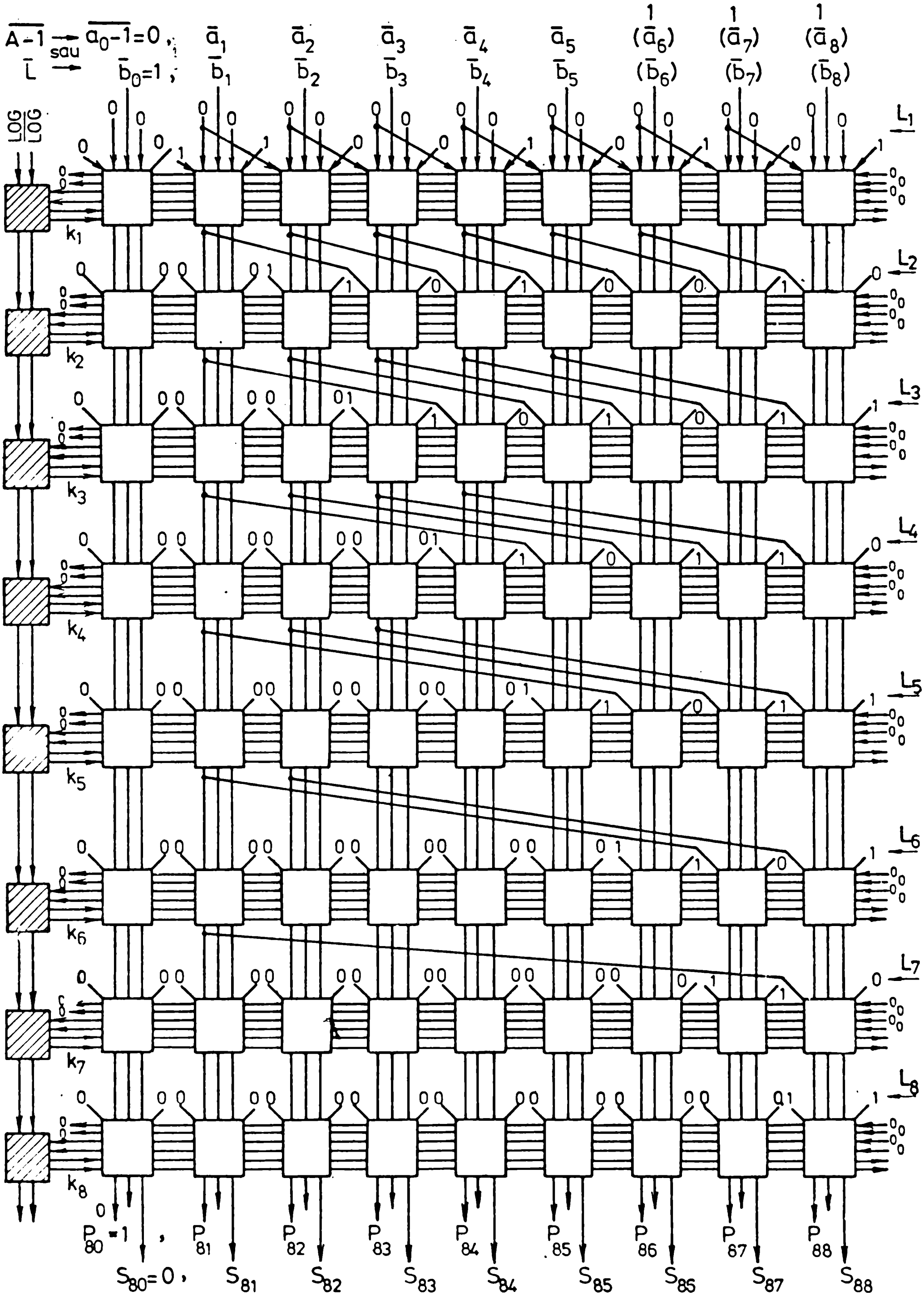


Fig.3-18 Structură logică celulară de generare a logaritmului şi antilogaritmului binar.

blocurile 6 iar antilogaritmul - la ieşirile de la blocurile 5 ale celulelor.

Intrucît produsul se obţine întotdeauna la aceeaşi ieşire pentru deplasarea produsului se folosesc aceleaşi legături diagonale atît la generarea logaritmului cît şi a antilogaritmului.

In fig.3-18 se prezintă structura logică celulară de generare a logaritmului (LOG=1) sau antilogaritmului ($\overline{LOG}$=1) al unui număr cu 8 biţi după virgulă,utilizînd celula din fig.3-17.

In scopul realizării celulelor sub formă de circuit integrat interesează,înafara complexităţii celulei,şi numărul de terminale necesar.In cazul în care unele bare ($\overline{k}_i,k_i,\overline{X}_j$) traversează celula rezultă,înafara terminalelor pentru alimentare cu tensiune continuă,următoarele cantităţi de terminale:

- o singură celulă într-o capsulă : 2o terminale,
- două celule în linie într-o capsulă : 28 terminale,
- patru celule în linie într-o capsulă : 44 terminale,
- două celule în linie şi două pe coloană într-o capsulă: 46 terminale.

In cazul în care barele $\overline{k}_i,k_i,\overline{X}_j$ nu traversează celula,înafara terminalelor de alimentare,sînt necesare următoarele cantităţi de terminale :

- o singură celulă într-o capsulă : 17 terminale,
- două celule în linie într-o capsulă : 24 terminale,
- patru celule în linie într-o capsulă : 38 terminale,
- două celule în linie şi două pe coloană într-o capsulă: 4o terminale.

Complexitatea unui grup de 4 celule nu constituie o problemă la realizarea sub formă de circuit integrat pe scară medie iar numărul de terminale necesar nu este exagerat de mare.Se poate aprecia ca realizabil un circuit integrat pe scară medie care să includă 4 celule în linie sau 2x2 celule într-o capsulă.In cazul utilizării grupajului de 2x2 celule pentru o structură logică celulară de generare a logaritmilor şi antilogaritmilor cu n par este necesară o cantitate de capsule:

$$N_{cs} = \frac{n(n+2)}{4} = \frac{n}{2}\left(\frac{n}{2} + 1\right) \qquad (3\text{-}65)$$

(structura va avea n+2 coloane din care n+1 în dreapta virgulei). Pentru n=28 rezultă un număr de capsule N_{cs}=21o.

La intrarea structurii din fig.3-18 (intrările verticale centrale ale celulelor din rîndul 1) se aplică în cazul operaţiei de logaritmare negaţiile biţilor numarului A-1 ce trebuie logaritmat

BUPT

sau-în cazul operației de antilogaritmare -negațiile numărului L ce trebuie antilogaritmat (deci $\bar{A}_j$ sau $\bar{L}_j$). Numărul aplicat la intrare poate avea m biți după virgulă (m≤n) sau n biți, după cum a rezultat în calcule în unitatea aritmetică.

La ieșirile din partea de jos ale structurii se obține mantisa logaritmului (ieșirile $S_{n,j}$) în cazul operației de logaritmare sau partea fracționară a antilogaritmului (ieșirile $P_{n,j}$) în cazul operației de antilogaritmare. Rezultatul se obține în ambele cazuri cu n biți după virgulă dar cu erori absolute negative, mai mici decît ΔA_p calculată în Capitolul II. Din aceștia se pot folosi în calcule fie m biți fie o cantitate mai mare.

Funcționarea structurii este identică cu funcționarea structurilor din fig.3-7 și 3-10, în cazul operației de logaritmare (LOG=1) sau a structurilor din fig.3-13 și 3-14, în cazul operației de antilogaritmare ($\overline{LOG}$=1).

Deoarece în fiecare celulă barele k_i și $\bar{k}_i$ comandă fiecare cîte două intrări, în cazul utilizării unor circuite NU-SI de putere cu sortanța 30 [69], pentru comanda barelor este necesară o refacere a semnalelor $\bar{k}_i$ și k_i după fiecare 14 celule ale unui rînd (fig.3-19). In fig.3-19 se prezintă o posibilitate de comandă a acestor bare pentru o srtuctură cu n=28; (n+1=29).

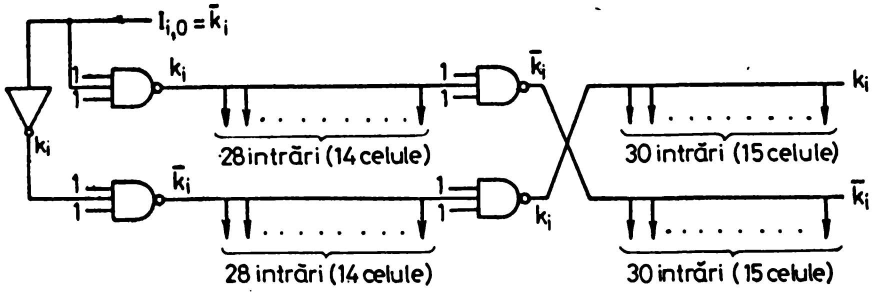


Fig.3-19 Realizarea semnalelor k_i și $\bar{k}_i$.

In această schemă funcția logică $\bar{k}_i$ de la intrare va fi :

$$\bar{k}_i = I_{i,0}LOG + I'_{i,0}\overline{LOG} \qquad (3-66)$$

și trebuie realizată înafara structurii logice. Astfel, în partea stîngă a structurii logice apare o celulă diferită de cea a structurii avînd schema din fig.3-20. Ea conține circuitul SI-SAU care realizează funcția (3-66) și circuitele NU-SI de putere pentru comanda barelor $\bar{k}_i$ și k_i.

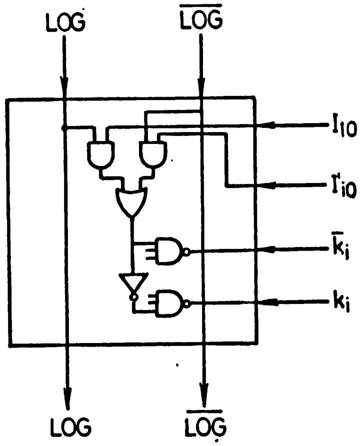


Fig.3-2o Celula pentru realizarea semnalelor k_i şi $\overline{k}_i$.

Intrucît o bară $\overline{X}_j$ comandă în fiecare celulă 4 intrări de circuite logice, dacă se foloseşte pentru comanda barei un circuit NU-ŞI de putere, cu sortanţa 30, atunci după fiecare 7 rînduri semnalul trebuie refăcut cu un invertor şi un nou circuit NU-ŞI de putere.

Aşa cum este de aşteptat, din cauza unui nivel logic suplimentar ce apare în celula din fig.3-2o, la realizarea funcţiei $\overline{k}_i$ (relaţia 3-66), durata operaţiilor de logaritmare şi antilogaritmare va creşte.Astfel în relaţia (3-37) timpul t_2 devine $t_2=3\ t_p$ şi relaţia ia forma:

$$t_{dfcmax} = N_{cit}t_p + (n_o+1)3\ t_p + n.3\ t_p \qquad (3\text{-}67)$$

Dacă numărul A-1 se aplică la intrarea structurii cu n biţi după virgulă atunci cazul cel mai defavorabil din punct de vedere al duratei descompunerii în factori apare cînd A-1= o,11...11 şi cînd N_{cit} este dat de relaţia (3-36) iar n_o= n-p-1.

Pentru acest caz, avînd n=28, p=4, n_o=23 N_{cit}=398 şi considerînd celulele realizate cu acelaşi tip de circuite ca şi mai înainte (t_p=6 ns) rezultă :

$$\underline{t_{dfcmax_n} = 554\ t_p = 3,324\ \mu s} \qquad (3\text{-}68)$$

In cazul cînd numărul ce se descompune, A-1, se aplică la intrarea structurii logice de descompunere în factori numai cu m biţi (n=m+4...,6), aşa cum s-a arătat în paragraful 3.3.1, cantitatea de celule parcurse de transport şi împrumut în propagarea acestora, N_{cit}, este dată de relaţia (3-4o) fiind mai mică decît în cazul anterior. Ea corespunde numărului A-1= o,olloo...oo. Pentru n=28, m=23, N_{cit}= 367, n_o= 22 şi t_p=6 ns se obţine :

$$\underline{t_{dfcmax_m} = 523\ t_p = 3,138\ \mu s}$$

Operaţia de antilogaritmare are o durată cu ceva mai mare decît cea de logaritmare deoarece în acest caz N_{cit} este mai mare. Relaţia (3-67) rămîne valabilă şi aici.

BUPT

Dacă logaritmul se aplică la intrarea structurii cu m biţi după virgulă atunci durata maximă a descompunerii apare în cazul în care $L=L_4$ (deci numai $k_4=1$) cînd N_{cit} este dat cu aproximaţie de relaţia (3-55). Pentru un n=28, $n_o=n-1=27$, $N_{cit}=417$ (determinat exact) şi $t_p=6$ ne rezultă :

$$t_{dtomax_n} = 585\ t_p = 3{,}51\ \mu s \qquad (3\text{-}70)$$

In cazul cînd logaritmul se aplică la intrarea structurii cu numai m biţi, durata maximă a descompunerii apare pentru $L=L_5+L_{26}$ (deci $k_5=k_{26}=1$) cînd N_{cit} este dat cu aproximaţie de relaţia (3-56). Pentru n=28, m=23, $n_o=26$, $N_{cit}=403$, $t_p=6$ ne rezultă o durată a descompunerii în termeni cu structurile comasate :

$$t_{dtomax_m} = 568\ t_p = 3{,}408\ \mu s \qquad (3\text{-}71)$$

O parte din celulele structurii logice celulare de la logaritmare şi antilogaritmare din fig.3-18 se pot simplifica ţinînd cont de următoarele :

1) nu toate celulele structurii se folosesc la descompunerea în factori şi la înmulţirea factorilor,

2) în celulele ce au aplicat la intrarea $P_{i-1,j-i}$ sau la intrarea $L_{i,j}$ un bit 0 se poate folosi în locul unui sumator complet - un semisumator deoarece intrarea respectivă se poate suprima,

3) la ieşirea celulelor din coloana 2^0 nu apare niciodată transport şi sumatoarele din aceste celule trebuie să realizeze doar suma logică - nu şi transportul.

Simplificările ce se pot aduce astfel circuitelor sînt însemnate în schimb ar fi necesară realizarea a 4 tipuri diferite de celule ceea ce, în cazul folosirii tehnologiei integrate, este mai costisitor. Prin simplificările de mai sus se reduce în mică măsură durata operaţiilor de logaritmare şi antilogaritmare.

3.3.4. Posibilităţi de îmbunătăţire a vitezei structurilor logice celulare de logaritmare şi antilogaritmare.

Anticiparea coeficientului de creştere la descompunerea în factori.

Analizîndu-se modul în care se desfăşoară descompunerea în factori după antilogaritmul propus (Tabelul 3-4) se pot face următoarele observaţii :

1. Pe măsură ce se avansează spre rîndurile de ordin superior produsul factorilor, obţinut la ieşirea unui rînd, conţine tot mai mulţi biţi identici cu biţii numărului A-1 ce se descompune, începînd din partea stîngă (coloana 2^0). Astfel, pentru exemplul

BUPT

considerat în Tabelul 3-4, produsul P_2-1 are primii cinci biţi identici cu primii cinci biţi ai numărului A-1. Pe baza acestei observaţii s-a delimitat printr-o linie verticală continuă în Tabelul 3-4 o "zonă de identitate". In zona de identitate biţii produsului de la ieşirea unui rînd se pot obţine direct din biţii produsului de la intrarea acelui rînd.

2. Bitul împrumutului din poziţia 2^0 este, într-un număr mare de cazuri, identic cu bitul împrumutului din poziţia aflată imediat în dreapta zonei de identitate (biţii încercuiţi). In restul cazurilor un bit 1 al produsului anterior deplasat, aflat în zona de identitate, cauzează apariţia unui împrumut în poziţia 2^0 a tabelului (biţii 1 încadraţi în patrate).

3. Ţinînd cont de posibilitatea determinării împrumutului din poziţia 2^0 pe baza observaţiei 2, în structura logică celulară de descompunere în factori sînt necesare, pentru un numar A dat, numai celulele corespunzătoare "zonei active" - încardată cu linie întreruptă în Tabelul 3-4. Pentru diferite numere A zona activă are configuraţii diferite astfel că interesează situaţiile limită în care numărul de celule necesar este maxim. Configuraţia limită a structurii, în cazul cel mai de-

Tabelul 3-4

	Poziţia	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	k_i
i	A-1	0	1	1	1	0	1	1	0	0	0	
	P_0-1	0	0	0	0	0	0	0	0	0	0	
1	$2^{-1}P_0$	0	1									1
	T_1	0	0									
	s_1	0	1									
	I_1	0	0									
	P_1-1	0	1									
2	$2^{-2}P_1$	0	0	1	1							1
	T_2	0	0	0	0							
	s_2	0	1	1	1							
	I_2	0	0	0	0							
	P_2-1	0	1	1	1							
3	$2^{-3}P_2$	0	0	0	1	$\boxed{1}$	1	1				0
	T_3	0	1	1	1	0	0	0				
	s_3	1	0	0	0	1	1	1				
	I_3	$\boxed{1}$	0	0	0	1	0	0				
	P_3-1	0	1	1	1	0	0	0				
4	$2^{-4}P_3$	0	0	0	0	$\boxed{1}$	1	1	1			0
	T_4	0	0	0	0	0	0	0	0			
	s_4	0	1	1	1	1	1	1	1			
	I_4	$\boxed{1}$	1	1	1	1	1	1	1			
	P_4-1	0	1	1	1	0	0	0	0			
5	$2^{-5}P_4$	0	0	0	0	0	1	1	1	1		0
	T_5	0	0	0	0	0	0	0	0	0		
	s_5	0	1	1	1	0	1	1	1	1		
	I_5	1	1	1	1	1	1	1	1	1		
	P_5-1	0	1	1	1	0	0	0	0	0		
6	$2^{-6}P_5$	0	0	0	0	0	0	1	1	1	1	1
	T_6	0	0	0	0	0	0	0	0	0	0	
	s_6	0	1	1	1	0	0	1	1	1	1	
	I_6	0	0	0	0	0	0	1	1	1	1	
	P_6-1	0	1	1	1	0	0	1	1	1	1	
7	$2^{-7}P_6$	0	0	0	0	0	0	0	1	1	1	1
	T_7	0	0	0	0	0	0	1	1	1	1	
	s_7	0	1	1	1	0	1	0	1	1	0	
	I_7	0	0	0	0	0	0	0	1	1	0	
	P_7-1	0	1	1	1	0	1	0	1	1	0	
8	$2^{-8}P_7$	0	0	0	0	0	0	0	0	1	1	0
	T_8	0	0	0	0	0	0	0	1	1	0	
	s_8	0	1	1	1	0	1	1	0	0	1	
	I_8	1	1	1	1	1	1	1	1	1	1	
	P_8-1	0	1	1	1	0	1	0	1	1	0	
9	$2^{-9}P_8$	0	0	0	0	0	0	0	0	0	1	1
	T_9	0	0	0	0	0	0	0	0	0	0	
	s_9	0	1	1	1	0	1	0	1	1	1	
	I_9	0	0	0	0	0	0	0	1	1	1	
	P_9-1	0	1	1	1	0	1	0	1	1	1	

favorabil în colţul din dreapta sus, se determină pe principiul arătat în fig.3-9 iar în partea stîngă corespunde cazului:

$$A - 1 = 0{,}01100\ldots00$$

Pentru n=9 structura logică celulară de descompunere în factori necesită numai celulele din fig.3-21. Celula din rîndul 1 se poate deasemenea înlătura deoarece, aşa cum s-a arătat în Capitolul II, întotdeauna $k_1 = a_1$ deci primul coeficient de creştere nu mai trebuie determinat.

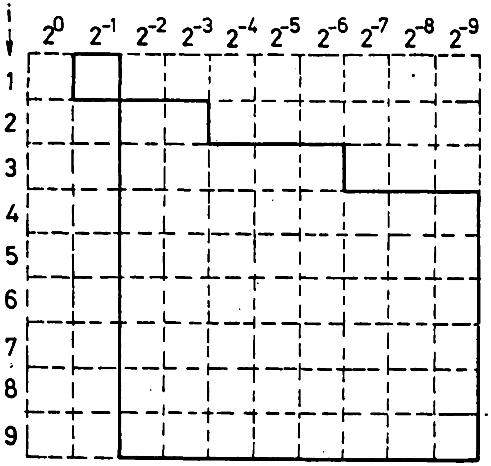


Fig.3-21 Zona activă a structurii.

Pe baza primelor două observaţii se poate stabili bitul împrumutului din poziţia 2^0 (care reprezintă negaţia coeficientului de creştere k_i) înainte ca împrumutul să se propage peste celulele unui rînd spre poziţia 2^0. Cu alte cuvinte, se poate anticipa coeficientul de creştere k_i prin introducerea în fiecare celulă a unor circuite logice suplimentare care să stabilească dacă celula se găseşte în zona de identitate şi să furnizeze un semnal $\bar{E}_{i,j}=1$ cînd în zona de identitate apare un bit al produsului deplasat sau cînd în celulele aflate imediat în dreapta zonei de identitate apare un împrumut.

Ieşirile $\bar{E}_{i,j}$ ale tuturor celulelor dintr-un rînd trebuie aplicate la intrările unui circuit logic SAU iar ieşirea acestuia se trimite prin toate celulele din acel rînd sub forma barelor k_i şi $\bar{E}_i$ pentru a comanda formarea produsului corect.

Se obţine astfel o nouă celulă pentru structura logică de descompunere în factori (fig.3-22). In cadrul celulei se realizează următoarele funcţii logice :

$$s_{i,j} = P_{i-1,j} \oplus P_{i-1,j-1} \oplus T_{i,j+1} \qquad (3\text{-}72)$$

$$T_{i,j} = P_{i-1,j}P_{i-1,j-1} + P_{i-1,j}T_{i,j+1} + P_{i-1,j-1}T_{i,j+1} \qquad (3\text{-}73)$$

$$I_{i,j} = \bar{A}_j(s_{i,j} + I_{i,j+1}) + s_{i,j}I_{i,j+1} \qquad (3\text{-}74)$$

$$ID_{i,j} = ID_{i,j-1}(P_{i-1,j}A_j + \bar{P}_{i-1,j}\bar{A}_j) + ID_{i-1,j} \qquad (3\text{-}75)$$

$$\bar{E}_{i,j} = ID_{i,j-1}I_{i,j} + ID_{i,j}P_{i-1,j-1} \qquad (3\text{-}76)$$

BUPT

$$P_{i,j} = \bar{k}_i P_{i-1} + k_i s_{i,j} + ID_{i,j} P_{i-1,j} \qquad (3\text{-}77)$$

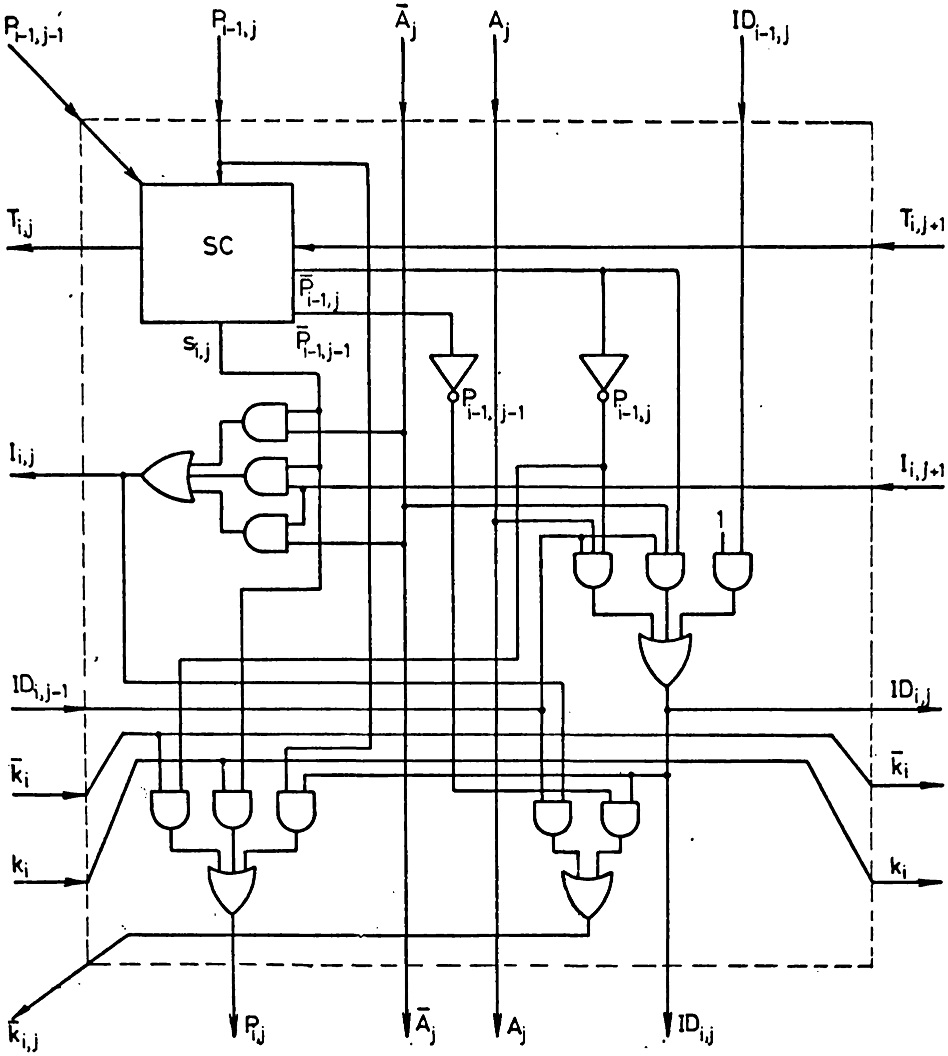


Fig.3-22. Celula structurii de descompunere în factori cu anticipare.

Funcțiile $s_{i,j}$, $T_{i,j}$ și $I_{i,j}$ sînt nemodificate față de cele de la structura logică celulară de descompunere în factori prezentată anterior (relațiile 3-29, 3-30, 3-31).

Funcția identitate $ID_{i,j}$ este egală cu 1 cînd în coloana anterioară ea este egală cu 1 iar biții P_{i-1} și A_j din coloana j coincid sau cînd în rîndul anterior i-1 ea este deja egală cu 1. La celulele din limita stîngă a structurii se va aplica la intrările $ID_{i,j-1}$ (laterale) bitul 1 iar la celulele din partea de sus a structurii la intrările $ID_{i-1,j}$ se va aplica bitul 0. In acest fel, zona de identitate, caracterizată prin $ID_{i,j}=1$ se va extinde treptat de la stînga spre dreapta şi de sus în jos pe măsură ce se determină o parte din biţii produsului în fiecare rînd.

Funcţia $\bar{k}_{i,j}$ realizează anticiparea coeficientului de creştere cu ajutorul variabilelor identitate, împrumut şi produs deplasat, aşa cum s-a arătat mai sus..

Pentru fiecare rînd al structurii mai trebuie introdus un circuit logic care să realizeze funcţiile :

$$\left.\begin{aligned} \bar{k}_i &= \bar{k}_{i,o}+ \bar{k}_{i,1}+ \bar{k}_{i,2}+\ldots+ \bar{k}_{i,n} \\ k_i &= \bar{\bar{k}}_i \end{aligned}\right\} \qquad (3\text{-}78)$$

şi să acţioneze prin circuite NU SI de putere barele k_i şi $\bar{k}_i$ ale celulelor rîndului i.

Ţinînd cont să ieşirea $P_{i,j}$ a unei celule constituie intrarea $P_{i-1,j}$ a celulei din coloana j a rîndului următor şi deasemenea intrarea $P_{i-1,j-i}$ a celulei din coloana j+i+1 a rîndului următor, din relaţiile (3-73)...(3-77) rezultă că o ieşire $P_{i,j}$ a unei celule trebuie să comande 14 intrări de circuite logice. Prin urmare ca ultim circuit în schema logică ce realizează funcţia $P_{i,j}$ ar trebui utilizat un circuit cu sortanţă mai mare sau egală cu 14, de exemplu, un circuit NU SI de putere. Aceasta ar impune realizarea funcţiei $P_{i,j}$ cu 2 nivele de circuite NU SI ori folosind un circuit SI-SAU [69], adică un singur nivel logic, s-ar obţine un timp de propagare de două ori mai mic. In acest din urmă caz, sortanţa circuitului SI-SAU fiind numai 10 este necesar să se distribuie ieşirea $P_{i,j}$ astfel : intrarea $P_{i-1,j}$ a unei celule să fie utilizată pentru comanda a cinci intrări de circuite logice ce realizează funcţiile $s_{i,j}$ şi $T_{i,j}$ unde timpul de propagare este un parametru însemnat iar intrarea $P_{i-1,j-i}$ a unei celule să fie utilizată deasemenea pentru cinci intrări în aceleaşi circuite. Pentru celelalte 4 intrări ce mai trebuie comandate se pot utiliza variabilele $P_{i-1,j}$ şi $P_{i-1,j-i}$ obţinute prin inversare din negaţiile $\bar{P}_{i-1,j}$ şi $\bar{P}_{i-1,j-i}$ (existente în circuitul logic al sumatorului) întrucît aici timpul t_{df} nu este mult influenţat (fig.3-22). Totuşi, ţinînd cont că în expresia sumei logice a trei biţi apar doi termeni de forma :

$$P_{i-1,j}\, P_{i-1,j-i}\, T_{i,j+1}$$

şi

$$P_{i-1,j}\, \bar{P}_{i-1,j-i}\, \bar{T}_{i,j+1}$$

în care nu poate fi simultan situaţia 011 - cînd la intrarea $P_{i-1,j}$ a circuitului NU-ŞI se consumă curent maxim - se deduce că la unul din circuitele NU-ŞI ce realizează aceşti termeni - cel ce are la intrare 000 - curentul consumat la intrarea $P_{i-1,j}$ este de aproape 3 ori mai redus (curentul se distribuie pe 3 intrări). Intrucît ieşirea $P_{i-1,j}$ a unei celule acţionează două sumatoare din celulele coloanelor j şi j+i se economiseşte suficient curent pentru ca ea să fie conectată la 11 intrări în loc de 10. Cea de a 11-a intrare se alege aceea de la circuitul ŞI-SAU al produsului $P_{i,j}$ care realizează transmiterea rapidă pe verticală a produsului $P_{i-1,j}$ cînd $ID_{i,j}$= 1, ceea ce este important la creşterea vitezei structurii logice de descompunere în factori.

In fig.3-23 şi 3-24 se prezintă două moduri de alcătuire

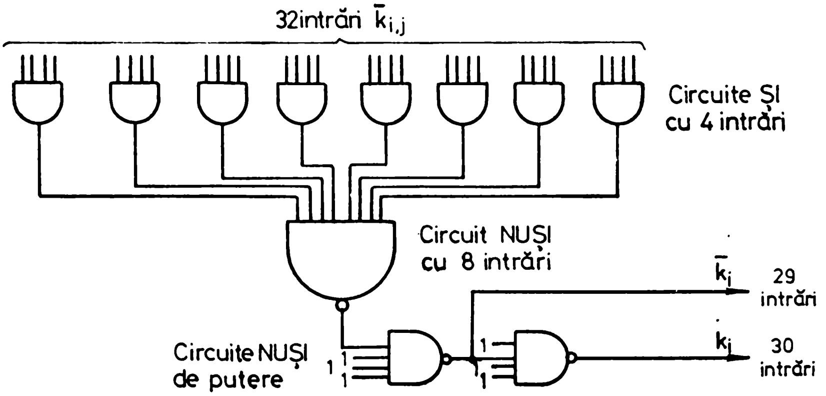


Fig.3-23. Circuit de anticipare

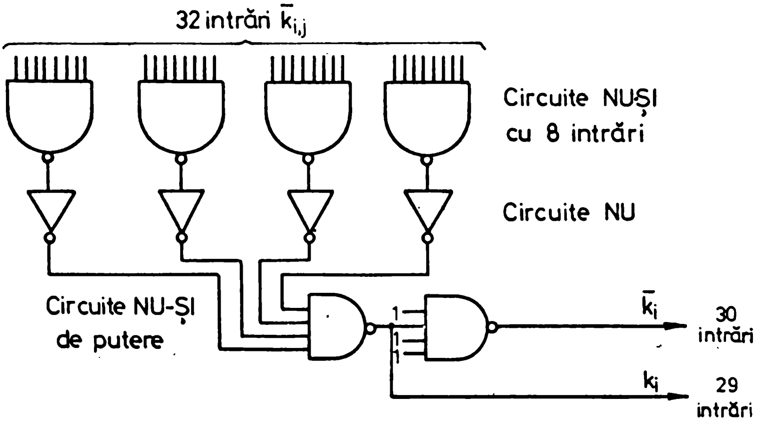


Fig. 3-24. Circuit de anticipare

BUPT

a circuitului ce realizează funcţiile (3-78) şi comandă barele k_i şi $\bar{k}_i$ ale unui rînd de celule.

Concepţia acestor celule s-a făcut considerînd circuitele integrate TTL rapide disponibile [69]. Ca număr de capsule necesare cele două scheme sînt practic echivalente.

Prin anticiparea coeficientului de creştere k_i în fiecare rînd al structurii, durata operaţiei de descompunere în factori se reduce întrucît nu mai este aşteptată propagarea unui împrumut peste celulele unui rînd pînă la ieşirea din celula coloanei 2^0 unde împrumutul devine variabila $\bar{k}_i$.

Utilizînd relaţia (3-37) modificată pentru acest caz se obţine durata operaţiei de descompunere în factori cu anticipare maximă :

$$t_{dfamax}= N_{cit}\cdot t_1+n_0 t_2+(n-1)(t_3+t_4) \qquad (3\text{-}79)$$

unde N_{cit} este numărul maxim de celule în care are loc generarea şi propagarea neîntreruptă a transportului şi împrumutului ; n_0 este numărul de rînduri cu $k_i = 0$; timpii t_1, t_3, t_4 au aceeaşi semnificaţie ca şi în relaţia (3-36) iar t_2 este timpul de propagare a variabilei $\bar{k}_i$ peste circuitele din fig.3-23 sau 3-24 şi care se poate considera $t_2 = 3\ t_p$.

Aici este necesar să se considere timpul de propagare pentru variabila $\bar{k}_i$ şi nu k_i deoarece în cazul cînd $k_i = 1$ această valoare este stabilită într-un rînd cu ceva mai înainte de terminarea operaţiilor în rîndurile anterioare unde $k_i = 0$. Deasemenea în cazul cel mai defavorabil din punct de vedere al duratei operaţiei de descompunere în factori produsul corect la ieşirea unui rînd este determinat de $\bar{k}_i$ deoarece în majoritatea rîndurilor $k_i = 0$. Astfel, pentru că în schema din fig.3-23 timpul de stabilire al lui $\bar{k}_i$ este mai mic decît al lui k_i, această schemă logică este cea mai indicată pentru obţinerea unei durate reduse a operaţiei de descompunere în factori.

Timpul necesar pentru stabilirea funcţiei logice $ID_{i,j}$ nu s-a inclus în relaţia (3-79) deoarece acest proces are loc în paralel cu calculul sumei.

Numărul maxim de celule N_{cit} în care are loc generarea şi propagarea consecutivă a transferului şi împrumutului apare în cazul cînd zona activă cuprinde un număr maxim de coloane iar majoritatea coeficienţilor de creştere sînt nuli. Această situaţie corespunde numărului

$$A-1 = 0,01100...00$$

cînd zona de identitate include doar coloanele din poziţiile $2^0, 2^{-1}$

BUPT

şi 2^{-2} (aceasta începînd cu rîndul 3). N_{cit} se poate calcula cu relaţia (3-40) din care se mai scade cantitatea $(3n_o-1)$ unde n_o este numărul de rînduri în care $k_1 = 0$:

$$N_{cit} = \frac{(n+2)(n-1)}{2} - (2^{p+1}+p+3n_o+1) \qquad (3-80)$$

Pentru n=28, p=4, n_o=22, N_{cit}=302 şi cu t_p= 6 ns rezultă:

$$t_{dfa_{max}} = 449\ t_p = 2{,}694\ \mu s \qquad (3-81)$$

Prin anticiparea valorilor coeficienţilor de creştere cu metoda prezentată se obţine o reducere de cca. 15,3% a duratei maxime a operaţiei de descompunere în factori în cazul structurilor necomasate (relaţia 3-39).

Intr-o măsură mult mai mare se reduce durata operaţiei de descompunere în factori pentru numere diferite de cel corespunzător cazului cel mai defavorabil. Durata minimă rămîne aceea din cazul cînd A-1 = 0,00...00.

După cum s-a arătat anterior întotdeauna $k_1 = a_1$ şi se poate vedea uşor că deasemenea întotdeauna p_{i1}, bitul produsului din toate rîndurile i = 1 ... n şi coloana 2^{-1} este identic cu a_1 - bitul din poziţia 2^{-1} al numărului ce se descompune. De aceea în structura logică celulară de descompunere în factori nu este necesar rîndul 1 de celule şi nici coloanele corespunzătoare poziţiilor 2^0 şi 2^{-1} (fig.3-21). Totuşi este necesar ca bitul p_{i1} al produsului din coloana 2^{-1} să fie deplasat şi el odată cu ceilalţi biţi ai produselor din fiecare rînd pentru a se aplica la intrările $P_{i-1,j-1}$ ale celulelor din coloanele i+1 (fig.3-25).

In fig.3-25 în celulele din stînga (de tip 2, diferit de cele ale structurii logice de descompunere) se află un circuit ca cel din fig.3-23 care realizează funcţia logică SAU a coeficienţilor de creştere din fiecare rînd al structurii. Restul celulelor (de tip 1) trebuie să aibă structura din fig.3-22. Pentru un n oarecare cantitatea celulelor de tip 1 necesare este :

$$N_{ca} = N_c - 2n$$

unde N_c este dat de relaţia (3-33). Pentru n=28 se obţine N_c= 644 celule.

Cantitatea celulelor de tip 2 este n-1. Intrucît realizarea celulelor de tip 2 într-o singură capsulă pune probleme privind cantitatea terminalelor şi lungimea conexiunilor cu celulele de tip 1, se pot distribui circuitele din celulele de tip 2 în apropierea unor grupuri de celule tip 1. In fig.3-25 ele s-au desenat

INSTITUTUL POLITEHNIC TIMIŞOARA CENTRALĂ

BUPT

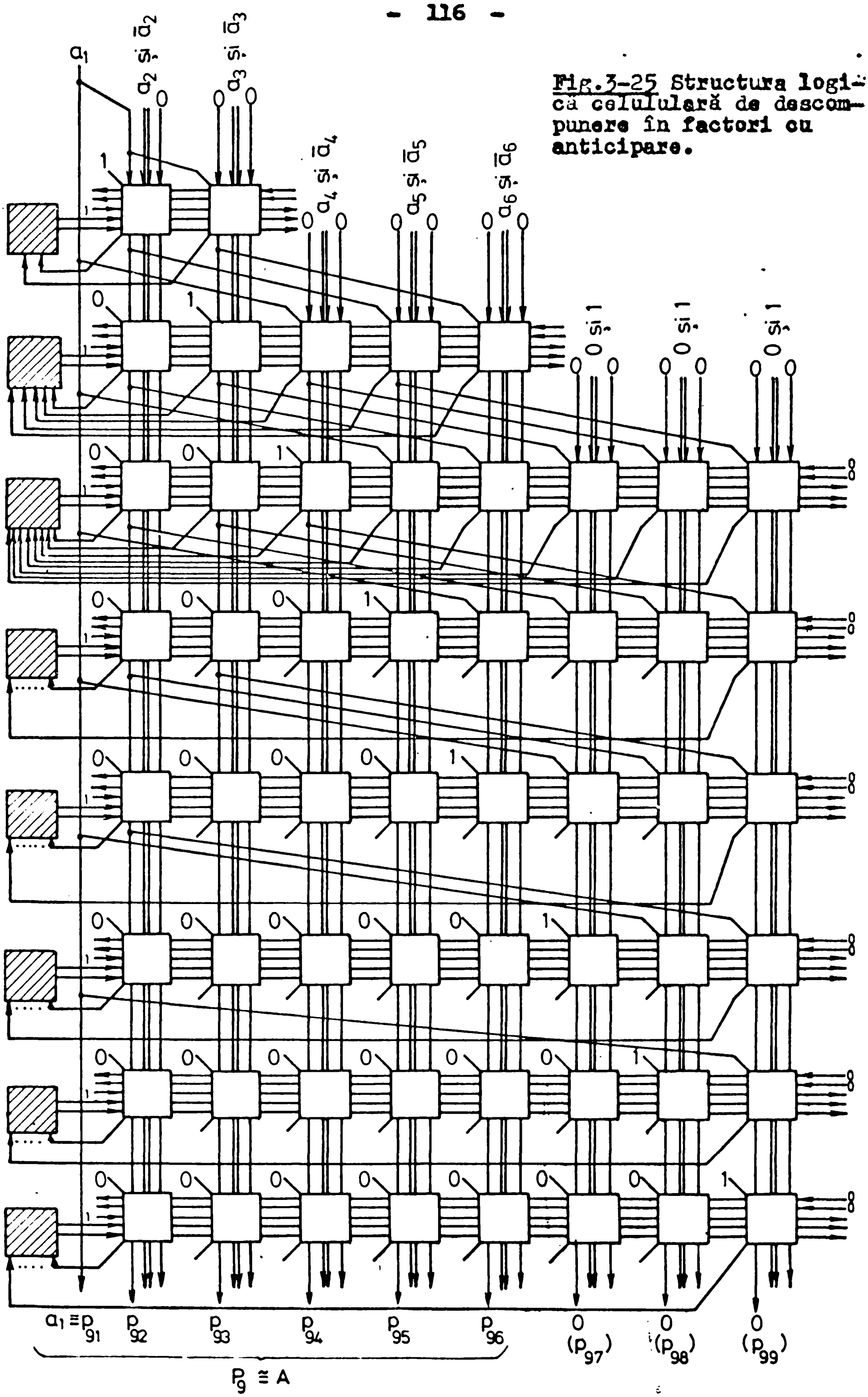


Fig.3-25 Structura logică celululară de descompunere în factori cu anticipare.

BUPT

în afara structurii din motive de simplitate a figurii.

In fig.3-25 bara verticală prin care se aplică bitul a_1 acţionează un număr mare de intrări şi anume, 5 la fiecare ramificaţie. Prin urmare sînt necesare din 6 în 6 rînduri circuite NU-ŞI de putere fiind necesar să se folosească tehnica din fig.3-19.

Anticiparea coeficientului de creştere la descompunerea în termeni.

Ţinînd cont de asemănarea mare dintre structura logică de descompunere în factori şi cea de descompunere în termeni se pot aplica şi la ultima din acestea observaţiile privind anticiparea împrumutului din coloana 2^0. Astfel, celula din fig.3-15 se poate realiza în forma din fig.3-26, asemănătoare cu celula din fig.3-22. Ieşirile $\bar{K}_{i,j}$ ale celulelor dintr-un rînd sînt conectate şi aici la un circuit logic (fig. 3-23) care stabileşte valoarea coeficientului de creştere k_i şi o transmite celulelor din rîndul respectiv.

In cadrul celulei se realizează funcţiile logice :

$$s'_{i,j} = S_{i-1,j} \oplus L_{i,j} \oplus T'_{i,j+1} \quad (3-82)$$

$$T'_{i,j} = S_{i-1,j}\, L_{i,j} + S_{i-1,j}\, T'_{i,j+1} + L_{i,j}\, T'_{i,j+1} \quad (3-83)$$

$$I'_{i,j} = \bar{L}_j(s'_{i,j} + I'_{i,j+1}) + s'_{i,j}\, I'_{i,j+1} \quad (3-84)$$

$$ID'_{i,j} = ID'_{i,j-1}(S_{i-1,j}L_j + \bar{S}_{i-1,j}\bar{L}_j) + ID'_{i-1,j} \quad (3-85)$$

$$\bar{K}_{i,j} = ID'_{i,j-1}I'_{i,j} + ID'_{i,j}L_{i,j} \quad (3-86)$$

$$S_{i,j} = \bar{k}_i S_{i-1,j} + k_i s'_{i,j} + ID'_{i,j}S_{i-1,j} \quad (3-87)$$

care se deosebesc de relaţiile (3-72)...(3-76) prin aceea că în locul lui $P_{i-1,j}$ intervine $S_{i-1,j}$, în locul lui $P_{i-1,j-i}$ intervine $L_{i,j}$, iar în locul lui A_j intervine aici L_j.

Prin folosirea celulei de mai sus în structura logică celulară de descompunere în termeni din fig.3-13 se poate renunţa la coloana de celule pentru bitul din faţa virgulei. Faţă de structura de descompunere în factori cu anticipare din fig.3-25 aici nu se mai pot elimina alte celule (fig.3-27) deoarece în cazul numărului L = o,1ooo... zona de identitate este minimă şi cuprinde numai coloana din poziţia 2^0 (din faţa virgulei).

Prin anticiparea împrumutului, deci şi a coeficientului de creştere k_i în fiecare rînd al structurii,durata operaţiei de descompunere în termeni se reduce.

Cazul cel mai defavorabil devine acum acela cînd se descompune în termeni logaritmul L=o,11oo...oo. Pentru N'_{cit} nu se poate găsi nici o relaţie de calcul. Dacă n=28 atunci N_{cit}= 259 iar durata operaţiei de descompunere în termeni este pentru acelaşi tip de

BUPT

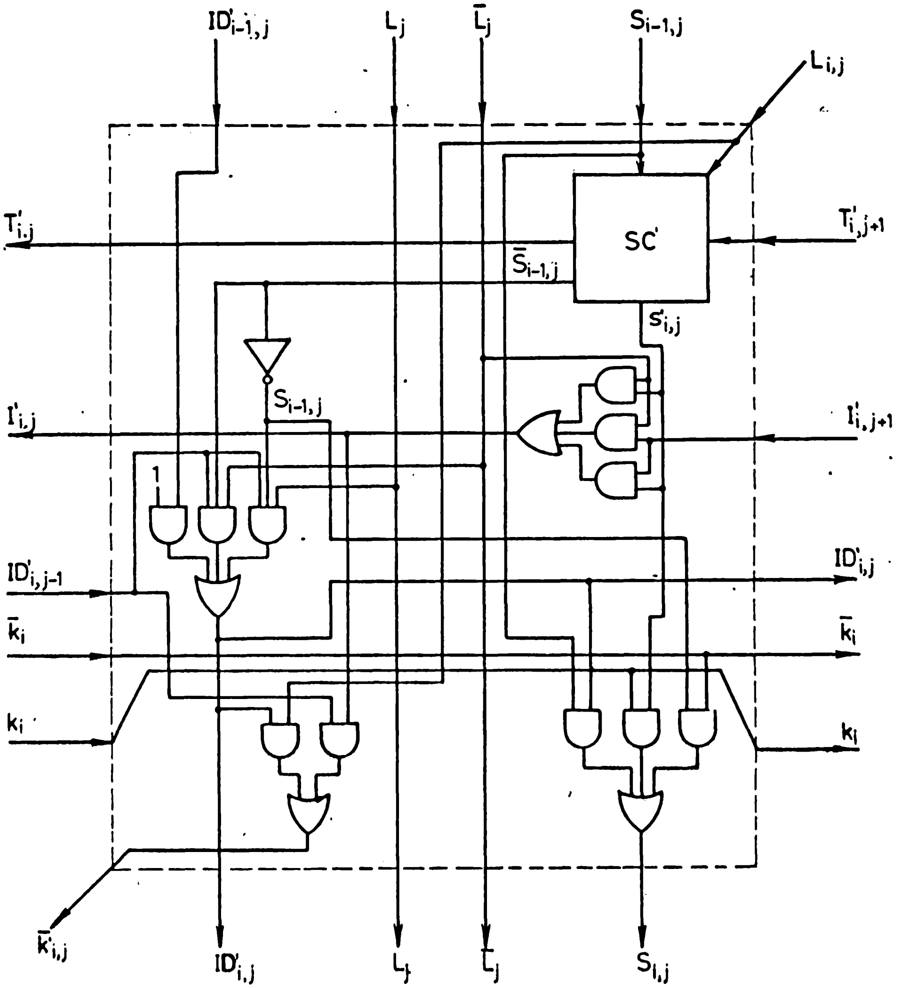


Fig.3-26. Celula structurii de descompunere în termeni cu anticipare.

circuite logice (t_p = 6 ns) :

$$\underline{t_{dtamax}} = N'_{cit} t_p + (n_o+1).3t_p + n(t_p + 2\ t_p) =$$

$$= \underline{379\ t_p = 2{,}274\ \mu s} \qquad (3\text{-}88).$$

Prin anticiparea valorii coeficientului de creştere durata maximă a operaţiei de descompunere în termeni scade cu cca 32% faţă de durata din structura fără anticipare (relaţia 3-58). Intr-o măsură şi mai mare se reduce durata operaţiei de descompunere în termeni pentru cazuri diferite de cel mai nefavorabil.

Durata minimă a operaţiei de descompunere în termeni rămîne aceeaşi ca la structura fără anticipare: t_{dtmin} = 0 şi corespunde

BUPT

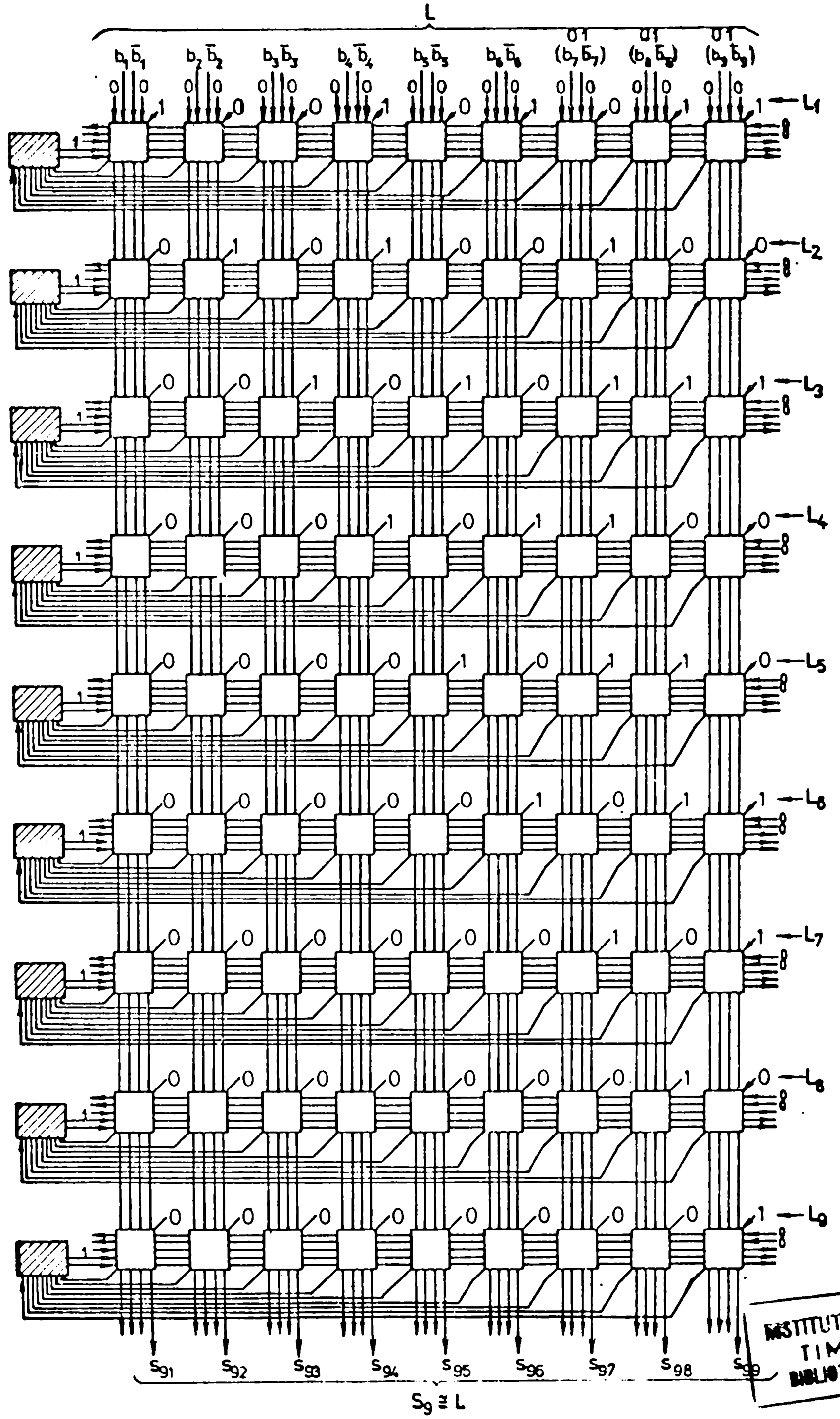


Fig.3-27 Structura logică celulară de descompunere în termeni cu anticipare.

INSTITUTUL POLITEHNIC TIMIŞOARA BIBLIOTECA CENTRALĂ

BUPT

cazului L = 0,00...00.

Structuri logice celulare de generare a logaritmilor şi antilogaritmilor cu anticipare şi comasare.

Aşa cum s-a arătat în paragraful 3.5.3 există posibilitatea de comasare a structurilor logice celulare de logaritmare şi antilogaritmare. Posibilitatea se păstrează şi dacă se foloseşte anticiparea coeficienţilor de creştere. Astfel, în fig.3-28 se prezintă schema bloc a unei celule obţinută prin comasarea celulelor din fig.3-22 şi 3-26. In această celulă, pentru a se putea deosebi operaţia de logaritmare, de antilogaritmare se introduc semnalele LOG şi $\overline{LOG}$ dintre care LOG=1 la logaritmare iar $\overline{LOG}$=1 la antilogaritmare. In lipsa acestor semnale în celule ar fi necesare circuite logice separate pentru funcţiile I şi I', ID şi ID', k_{ij} şi k'_{ij} şi bare separate de intrare şi ieşire din celulă pentru aceste funcţii precum şi circuite logice ca cel din fig. 3-23 separate pentru logaritmare şi antilogaritmare.

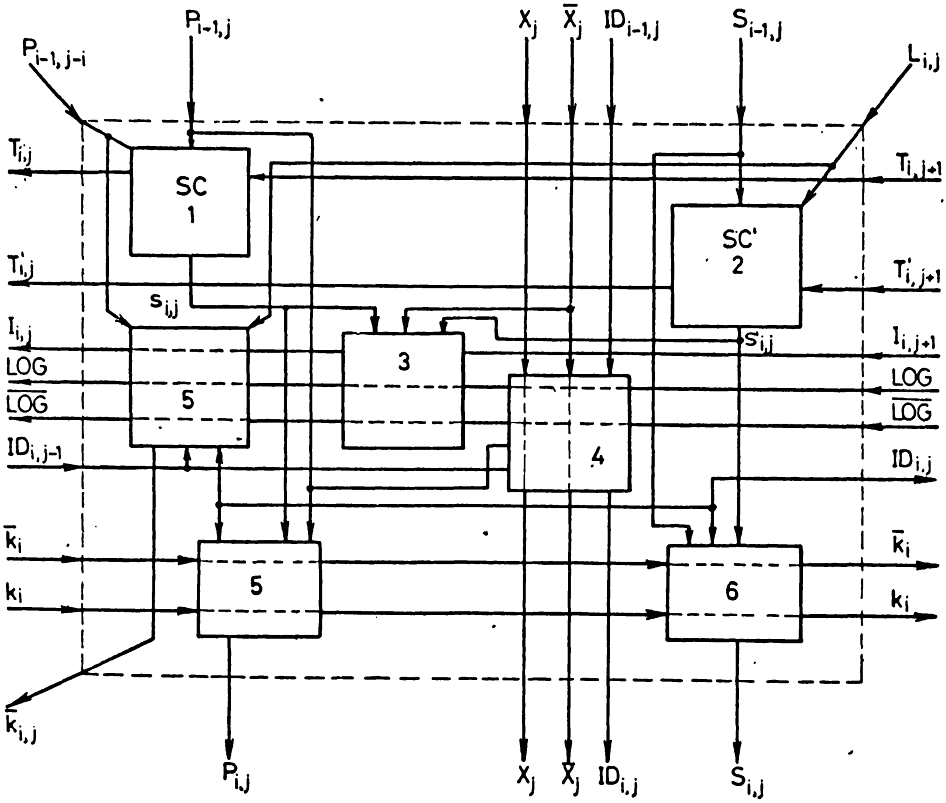


Fig.3-28. Celula structurii de generare a logaritmilor şi antilogaritmilor cu anticipare şi comasare.

BUPT

In aceasta blocurile logice 3,4 şi 5 realizează funcţiile:

$$I_{i,j} = X_j\left[(s_{i,j}LOG+s'_{i,j}\overline{LOG})+I_{i,j+1}\right]+(s_{i,j}LOG+s_{i,j}\overline{LOG})I_{i,j+1} \quad ,(3\text{-}89)$$

$$ID_{i,j}= ID_{i,j-1}\left[(P_{i-1,j}LOG+S_{i-1,j}\overline{LOG})X_j+(\overline{P}_{i-1,j}LOG+\overline{S}_{i-1,j}\overline{LOG})\overline{X}_j\right]+ + ID_{i-1,j} \quad (3\text{-}90)$$

$$E_{i,j} = ID_{i,j-1}I_{i,j}+ ID_{i,j}(P_{i-1,j-i}LOG+L_{i,j}\overline{LOG}) \quad (3\text{-}91)$$

Dacă se utilizează celula din fig.3-26 atunci durata operaţiilor de generare a logaritmului şi antilogaritmului se modifică faţă de structurile necomasate. In drumul de propagare consecutivă a transportului şi împrumutului apare un nivel logic suplimentar şi anume, la stabilirea sumei care se compară dintre $s_{i,j}$ şi $s'_{i,j}$ (relaţia 3-89). De aceea aici $t_4= 3t_p$ şi durata maximă a operaţiei de descompunere în factori devine (relaţia 3-88) :

$$t_{dfacmax} = N_{cit}t_p+(n_o+1)3t_p+ n(t_p+3t_p) \quad (3\text{-}92)$$

Pentru n=28 rezultă $N_{cit}= 302$ şi $n_o=22$. Pentru $t_p= 6$ ns se obţine $t_{dfacmax} = 476\ t_p= 2,856\ \mu s$ adică o reducere cu cca.14% a duratei maxime a operaţiei din structurile logice celulare comasate fără anticipare (relaţia 3-68).

In mod asemănător, durata maximă a operaţiei de descompunere în termeni în cazul structurilor comasate cu anticipare va fi dată tot de relaţia (3-92) în care se introduce numărul N_{cit} corespunzător. Pentru n=28, n_o=11 şi N_{cit}=259 cu t_p=6 ns rezultă:

$$\underline{t_{dtacmax} = 407\ t_p = 2,442\ \mu s} \quad (3\text{-}93)$$

adică se obţine o reducere a duratei maxime a operaţiei de descompunere în termeni de cca.30,4% faţă de durata din cazul structurilor logice comasate fără anticipare (relaţia 3-70).

Celula din fig.3-28 necesită un număr de 29 terminale, înafara celor de alimentare cu tensiune continuă, dacă pentru barele ce traversează celula se prevăd şi intrări şi ieşiri sau 23 de terminale, dacă pentru barele amintite se prevăd numai intrări. Un grup de 2 celule în linie necesită 42 terminale cînd pentru barele ce traversează celula se prevăd şi intrări şi ieşiri sau 34 terminale în caz contrar. Pentru un grup de 2x2 celule rezultă în cele două situaţii 66 respectiv 54 terminale iar pentru un grup de 4 celule în linie - 70 respectiv 58 terminale. Există deci posibilitatea integrării unei singure celule sau a unui grup de 2 celule în linie.

In fig.3-29 se prezintă o structură logică celulară de logaritmare şi antilogaritmare cu anticiparea coeficienţilor de

BUPT

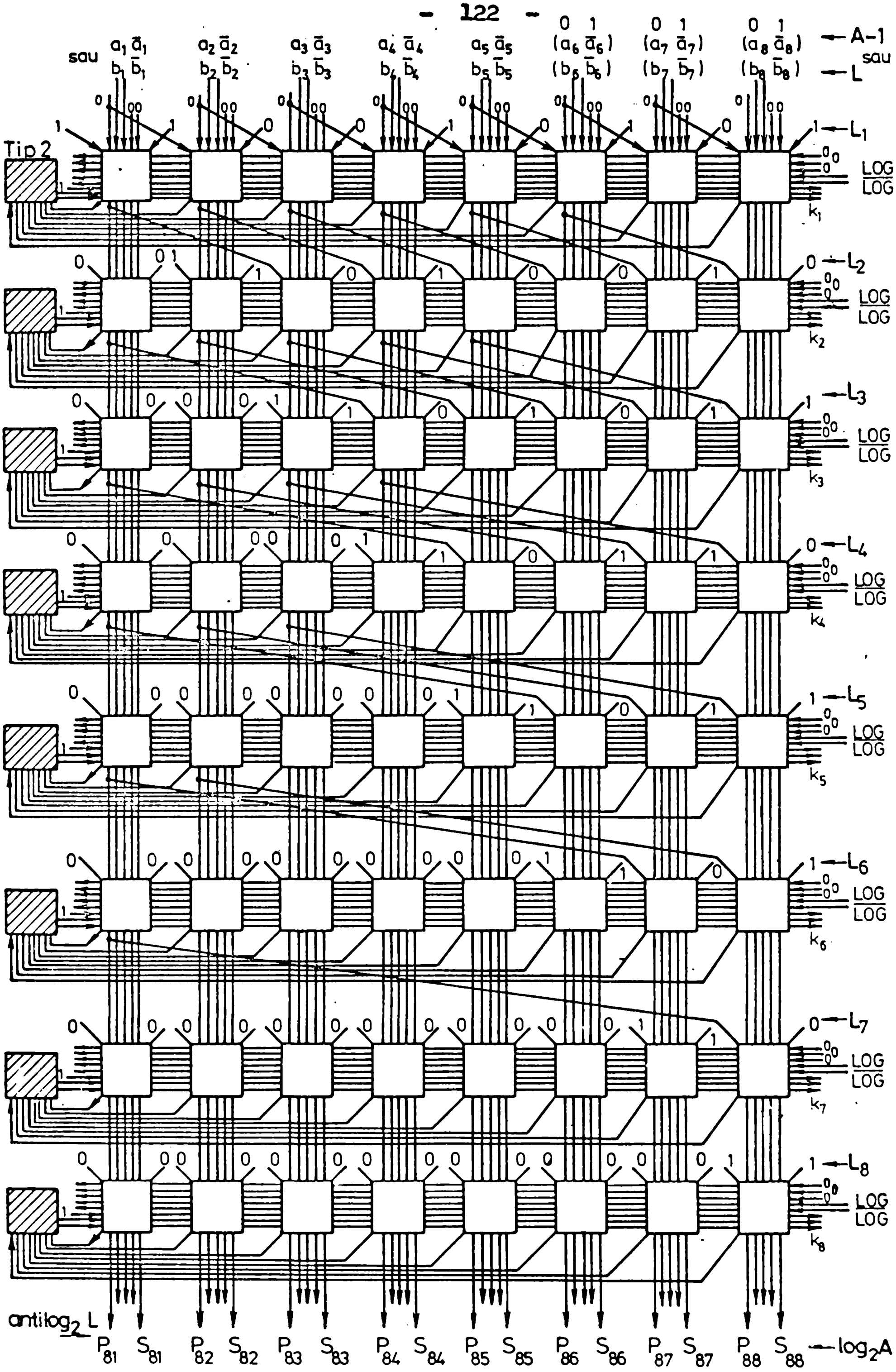


Fig.3-29.Structura logică celulară de generare a logaritmilor şi antilogaritmilor binari cu anticipare şi comasare.

BUPT

creştere pentru n=8 biţi. Deşi o parte din celule nu sînt necesare pentru descompunerea în factori la logaritmare (fig.3-25) şi pentru înmulţirea factorilor la antilogaritmare, totuşi celulele respective sînt necesare la descompunerea în termeni sau la însumarea termenilor. Astfel structura logică celulară trebuie să cuprindă n^2 celule de tip 1 şi n celule de tip 2. Celulele de tip 2 din coloana din stînga realizează funcţia logică SAU a coeficienţilor de creştere $k_{i,j}$ din rîndul respectiv al structurii conform schemei logice din fig.3-23. Celulele de tip 1 au schema din fig.3-28. In structură nu mai este necesară coloana pentru poziţia binară din stînga virgulei.

Realizarea operaţiilor de descompunere în factori şi în termeni în mod asincron.

După cum s-a arătat în paragrafele anterioare, durata operaţiei de generare a logaritmului sau antilogaritmului este cuprinsă între 0 şi o valoare maximă t_{dfmax} sau t_{dtmax}, funcţie de valoarea numerelor cu care se operează. Dacă operaţiile se fac în mod sincron atunci este necesar să se reserve pentru o operaţie un timp $t \geqslant t_{dtmax}$ sau t_{dtmax} indiferent de valoarea numărului (aşa cum s-a arătat, un transport lung în ultimul rînd al structurii de însumare a termenilor sau al structurii de înmulţire a factorilor nu apare în cazul cînd durata descompunerii este maximă deci durata operaţiei este stabilită de durata descompunerii).

Pentru a se realiza o economie însemnată de timp la calculul logaritmilor şi antilogaritmilor este necesar ca operaţiile de descompunere - care stabilesc durata operaţiilor de logaritmare sau antilogaritmare - să se efectueze în mod asincron. Pentru aceasta este necesar să se stabilească momentul în care descompunerea s-a terminat (au fost determinaţi coeficienţii de creştere k_i).

O posibilitate de stabilire a momentului terminării descompunerii o constituie compararea numărului ce se descompune (A sau L) cu cantitatea ce îl aproximează după ultimul rînd al structurii, adică :

- cu produsul P_n la descompunerea în factori,
- cu suma S_n la descompunerea în termeni.

In momentul în care cele două numere ce se compară devin egale, operaţia de descompunere este terminată.

In cazul structurilor logice celulare cu anticiparea coeficientului de creştere se poate utiliza semnalul de identitate ID în acest scop. Totuşi sînt necesare circuite suplimentare pentru realizarea funcţiei ID după ultimul rînd al structurii întrucît

BUPT

produsul P_n şi suma S_n sînt abia atunci disponibile.

Singura dificultate ce apare la sesisarea terminării descompunerii este cauzată de eroarea ce apare la descompunere şi care - aşa cum s-a arătat în Capitolul II - face ca :

$$P_n \leqslant A \qquad \text{şi} \qquad S_n \leqslant L$$

Eroarea aceasta, întotdeauna negativă, se datoreşte folosirii unui număr finit de biţi în factorii $(1+2^{-i})$ şi în termenii $L_i=\log_2(1+2^{-i})$ fapt care conduce la "scăparea" unor factori sau termeni din descompunere, aşa cum s-a arătat în Capitolul II.

Astfel, la descompunerea în factori apare o eroare :

$$-P_n < P_{\frac{n}{2}+1}\left[1+2^{-(\frac{n}{2}+1)}\right](1+2^{-n}) - P_{\frac{n}{2}+1}\prod_{i=\frac{n}{2}+2}^{i=n}(1+2^{-i}) <$$

$$< \left\{\left[1+2^{-(\frac{n}{2}+1)}\right](1+2^{-n}) - (1+\sum_{i=\frac{n}{2}+2}^{i=n} 2^{-i})\right\} \cong$$

$$\cong 2\left[(1+2^{-(\frac{n}{2}+1)}+2^{-n}) - (1+\sum_{i=\frac{n}{2}+2}^{i=n} 2^{-i})\right] = 2^{-(n-2)} \tag{3-94}$$

care pentru n=28 biţi are valoarea $\Delta P_{28} < 2^{-26}$.

Pentru descompunerea în termeni eroarea este :

$$\Delta S_n < L_{\frac{n}{2}+1} + L_{n+2-m} + L_{n+3-m} - \sum_{i=\frac{n}{2}+2}^{i=n} L_i = n_1 2^{-n} + L_{n+2-m} + L_{n+3-m} \tag{3-95}$$

unde n_1 este numărul de biţi 1 în ultima poziţie a constantelor L_i pentru $i=\frac{n}{2}+1\ldots n$ iar m este numărul întreg maxim pentru care $2^m \leqslant n$. Pentru n = 28 rezultă $n_1 = 7$, $m = 4$ şi deci :

$$\Delta S_{28} \leqslant 2^{-25} + 2^{-26} + 2^{-27} \tag{3-96}$$

Deci, pentru ca numerele ce se compară să coincidă (P_n cu A sau S_n cu L) în scopul determinării sfîrşitului operaţiei de descompunere, este necesar ca la P_n şi la S_n să se adune erorile maxime ΔP_n şi ΔS_n de mai sus. Dacă numărul A sau L se aduce cu m biţi după virgulă (m poate fi cel mult egal cu numărul de biţi ai lui P_n sau S_n neafectaţi de eroarea ΔP_n sau ΔS_n) la intrarea structurii logice restul de biţi pînă la n sînt stabiliţi automat la valoarea zero. Cînd după descompunere rezultă P_n=A şi S_n=L atunci ultimii n-m biţi ai lui P_n şi S_n sînt deasemenea nulă şi adunarea erorii ΔP_n sau ΔS_n, care se face în ultimele n-m poziţii, nu

BUPT

schimbă primele m poziţii de după virgulă ale lui P_n sau S_n deci identitatea $P_n = A$ sau $S_n = L$ se poate realiza pe primii m biţi după virgulă ceea ce este suficient pentru stabilirea momentului terminării descompunerii. Dacă însă la intrarea structurii numărul A sau L se aplică cu n biţi situaţia de mai sus nu se păstrează, adică nu se poate realiza identitatea pentru m biţi după virgulă prin adunarea corecţiei.

Dacă după descompunere rezultă $P_n < A$ sau $S_n < L$ (eroarea este întotdeauna negativă) atunci o parte din ultimii n-m biţi ai lui P_n sau S_n sînt diferiţi de 0 iar identitatea P_n cu A sau S_n cu L pe primii m biţi după virgulă lipseşte, diferenţa dintre aceste numere fiind 2^{-m} pentru primii m biţi. Adunarea erorii maxime ΔP_n sau ΔS_n în ultimele n-m poziţii ale lui P_n sau S_n cauzează în acest caz un transport spre poziţia 2^{-m} şi se stabileşte identitatea $P_n = A$ sau $S_n = L$ pe primii m biţi. O parte din ultimii n-m biţi ai lui P_n sau S_n pot să rămînă în continuare diferiţi de zero, ceea ce nu deranjează deloc.

In acest mod, prin adunarea erorii maxime de la descompunere la P_n sau S_n se poate realiza întotdeauna identitatea $P_n = A$ sau $S_n = L$ la terminarea descompunerii <u>în cazul cînd A sau L se aplică la intrare cu numai m biţi.</u> Prin aceasta se stabileşte momentul terminării descompunerii.

Intrucît acum este posibil ca după terminarea descompunerii în factori sau termeni să apară un transport de lungime maximă în ultimul rînd al structurii logice celulare de însumare a termenilor sau de înmulţire a factorilor, momentul terminării operaţiei de logaritmare sau antilogaritmare trebuie să fie considerat (ţinînd cont şi de timpul de formare a sumei) :

$$t_{log} = t_{dfas} + (n+2)t_p \qquad (3\text{-}97)$$

sau

$$t_{antilog} = t_{dtas} + (n+2)t_p \qquad (3\text{-}98)$$

unde t_p este timpul de propagare mediu pe un nivel logic al transportului iar t_{dfas} sau t_{dtas} este stabilit de momentul identităţii $P_n = A$ sau $S_n = L$ în structura asincronă.

In cazul cînd structurile logice de logaritmare şi antilogaritmare sînt comasate (fig.3-19 şi 3-29) la ieşirile P_n se obţine la logaritmare produsul $P_n \cong A$ iar la antilogaritmare-antilogaritmul $P_n \cong \text{antilog}_2 L$. In mod asemănător, la ieşirile S_n se obţine la logaritmare logaritmul $S_n \cong \log_2 A$ iar la antilogaritmare suma $S_n \cong L$. De aceea este necesar ca efectuarea corecţiei de mai sus asupra lui P_n să se facă numai la logaritmare iar asupra lui S_n -numai la an-

BUPT

tilogaritmare. Dacă se ţine însă cont că eroarea de la calculul logaritmului şi antilogaritmului - aşa cum s-a arătat în Capitolul II - se poate reduce la jumătate prin adunarea la S_n (în cazul logaritmării) sau la P_n (în cazul antilogaritmării) a unei corecţii egală cu :

$$\frac{\Delta L_p}{2} \quad \text{sau} \quad \frac{\Delta A_p}{2}$$

atunci ar fi necesare în ansamblu următoarele corecţii (pentru n=28 de exemplu) :

- la calculul logaritmului :
  a) corecţia lui P_n pentru realizarea identităţii $P_n \equiv A$ pe primii 24 biţi după virgulă ($m \leqslant 24$) care poate fi cuprinsă în domeniul:

$$2^{-26} \leqslant cor_1' < 2^{-24} \qquad (3\text{-}99)$$

  (oloo ... 1111 în binar în poziţiile $2^{-25} \ldots 2^{-28}$) întrucît după corecţie nu mai contează conţinutul ultimilor 4 biţi.

  b) corecţia lui S_n pentru reducerea la jumătate a erorii absolute maxime a logaritmului, cu cantitatea :

$$cor_2' = 2^{-25} + 2^{-27} + 2^{-28} \qquad (3\text{-}1oo)$$

  (loll în binar în poziţiile $2^{-25} \ldots 2^{-28}$)

- la calculul antilogaritmului :
  a) corecţia lui S_n pentru realizarea identităţii $S_n \equiv L$ pe primii 24 biţi după virgulă, care poate fi cuprinsă în domeniul :

$$2^{-25} + 2^{-26} + 2^{-27} \leqslant cor_2'' < 2^{-24} \qquad (3\text{-}1o1)$$

  (llo ... 1111 în binar în poziţiile $2^{-25} \ldots 2^{28}$)

  b) corecţia lui P_n pentru reducerea la jumătate a erorii maxime a antilogaritmului cu cantitatea :

$$cor_1'' = 2^{-25} + 2^{-27} + 2^{-28} \qquad (3\text{-}1o2)$$

  (loll în binar în poziţiile $2^{-25} \ldots 2^{-28}$)

După cum se observă, corecţiile lui P_n în cele două cazuri pot fi făcute identice : cor_1= loll în timp ce corecţiile lui S_n (cor_2) nu se pot face identice şi deci corecţiile cor_2' şi cor_2'' trebuie realizate, folosind un circuit logic, în funcţie de semnalele LOG şi $\overline{LOG}$.

Efectuarea corecţiilor asupra logaritmului şi antilogaritmului are însă urmări de care trebuie ţinut cont la concepţia dis-

BUPT

pozitivului aritmetic. Astfel, prin adunarea corecţiei $\Delta L_p/2$ sau $\Delta A_p/2$ la rezultatul aproximativ, eroarea absolută finală poate fi cuprinsă între

$$-\frac{\Delta L_p}{2} \ldots + \frac{\Delta L_p}{2} \quad \text{sau} \quad -\frac{\Delta A_p}{2} \ldots + \frac{\Delta A_p}{2}$$

deci poate fi negativă sau pozitivă. Pentru rezultate apropiate de 1 în cazul logaritmului sau 2 în cazul antilogaritmului este posibil să apară în urma corecţiei un logaritm mai mare decît 1 sau un antilogaritm mai mare decît 2 care trebuiesc reduşi în domeniul stabilit iniţial prin corectarea caracteristicii -în cazul logaritmului- sau deplasare la dreapta şi corectarea exponentului - în cazul antilogaritmului.

Dacă precizia de calcul se consideră satisfăcătoare se poate renunţa la corecţia $\Delta L_p/2$ sau $\Delta A_p/2$.

In cele ce urmează se prezintă circuitele pentru stabilirea identităţilor $P_n \equiv A$ şi $S_n \equiv L$ cu efectuarea numai a corecţiilor necesare realizării identităţii pe m biţi (se consideră situaţia concretă cu n=28 şi $m \leq 24$ biţi) în cazul structurilor comasate cu anticiparea coeficienţilor de creştere.

In acest scop se poate utiliza un rînd suplimentar de celule la ieşirea structurii din fig.3-29. Celulele au forma dată în fig.3-30 (tip 3) şi 3-31 (tip 4).

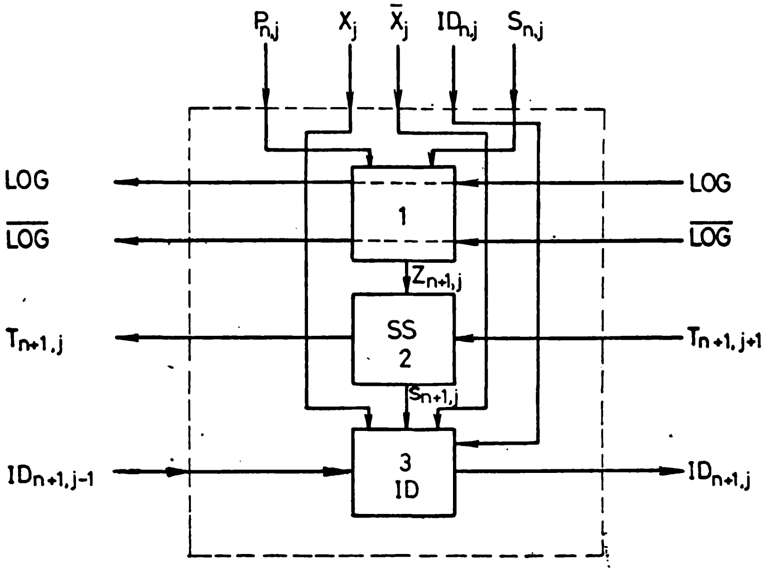


Fig.3-30. Celula din rîndul suplimentar pentru realizarea identităţii.

Blocul 1 din fig.3-3o realizează funcţia logică :

$$Z_{n+1,j} = P_{n,j}LOG + S_{n,j}\overline{LOG} \qquad (3\text{-}1o3)$$

Blocul 2 este un semisumator şi realizează semisuma şi transportul:

$$s_{n+1,j} = Z_{n+1,j} \oplus T_{n+1,j+1} \qquad (3\text{-}1o4)$$

$$T_{n+1,j} = Z_{n+1,j}\, T_{n+1,j+1} \qquad (3\text{-}1o5)$$

iar blocul 3 realizează funcţia logică :

$$ID_{n+1,j} = ID_{n+1,j-1}(Z_{n+1,j}X_j + \overline{Z}_{n+1,j}\overline{X}_j) + ID_{n,j} \qquad (3\text{-}1o6)$$

Intrucît corecţia se adună numai la ultimele 4 poziţii ale lui P_n sau S_n, celulele din poziţiile 2^{-1}:..2^{-24} necesită numai semisumatoare şi pot fi de tipul celei din fig.3-3o.(tip 3). Pentru ultimele 4 poziţii, unde se adună corecţia sînt necesare celule de tipul celei din fig.3-31 (tip 4).

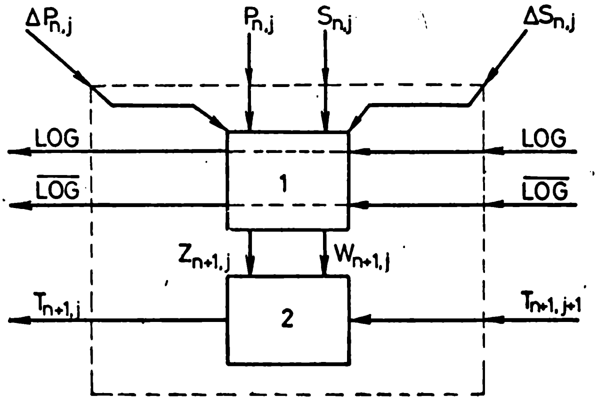


Fig.3-31. Celula din poziţiile 25-28 ale rîndului suplimentar.

Blocul 1 din fig.3-31 realizează funcţia $Z_{n+1,j}$ (3-1o4) şi funcţia :

$$W_{n+1,j} = \Delta P_{n,j}LOG + \Delta S_{n,j}\overline{LOG} \qquad (3\text{-}1o7)$$

iar blocul 2 realizează transportul pentru trei intrări :

$$T_{n+1,j} = Z_{n+1,j}W_{n+1,j} + Z_{n+1,j}T_{n+1,j+1} + W_{n+1,j}T_{n+1,j+1} \qquad (3\text{-}1o8)$$

Rîndul suplimentar de celule are forma din fig.3-32.

După cum se observă, bitul corecţiei adunat la ultima celulă din dreapta este nul şi celula nu mai este necesară. Deasemenea, celulele de tip 2 rămase se pot simplifica întrucît toate au la intrarea ΔS_n bitul 1 iar la intrările diagonale ΔP_n se pot aplica deasemenea numai biţi 1. Pe baza acestor observaţii întregul circuit cu celule de tip 4 se poate înlocui cu un circuit simplu care să in-

BUPT

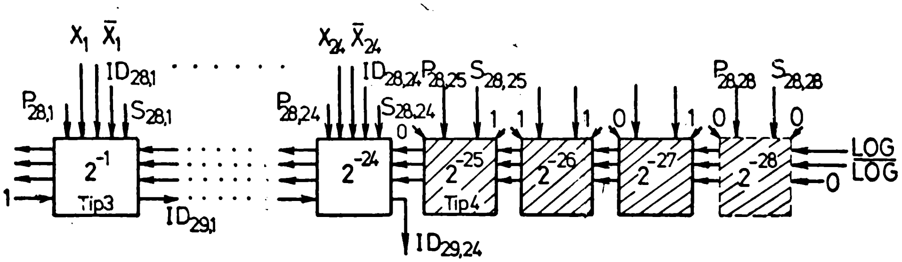


Fig.3-32. Rîndul de celule suplimentar pentru realizarea identităţii.

troducă un transport 1 la intrarea celulei din poziţia 2^{-24} dacă unul din biţii din poziţiile $2^{-25}...2^{-28}$ ai produsului P_n sau sumei S_n este 1.

La ieşirea $ID_{29,24}$ (fig.3-32) se obţine semnalul de terminare a descompunerii în factori sau termeni.

Dacă este necesară şi corecţia rezultatului (logaritmului sau antilogaritmului) în scopul reducerii erorii maxime la jumătate, atunci trebuie să se utilizeze în rîndul suplimentar 24 de celule de tip 3 prezentate în fig.3-33 şi 4 celule de tip 4 prezentate în fig.3-34.

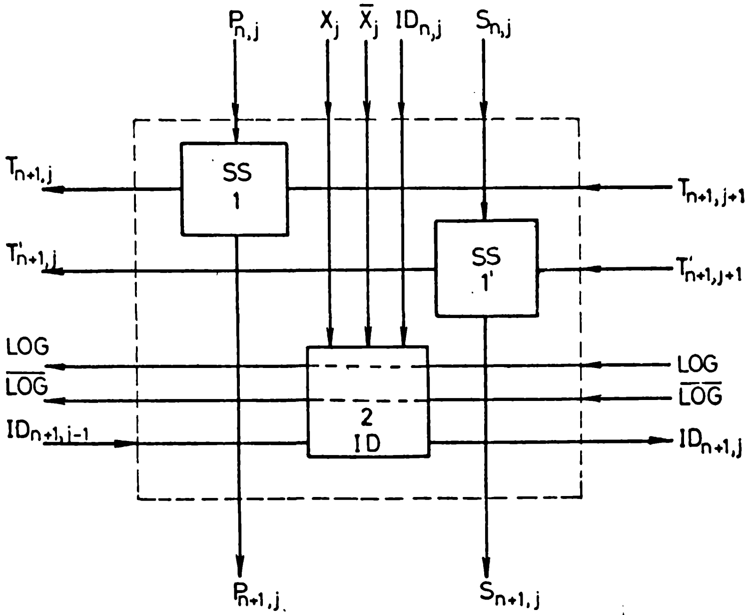


Fig.3-33. Celula din rîndul suplimentar pentru realizarea identităţii şi corecţiei.

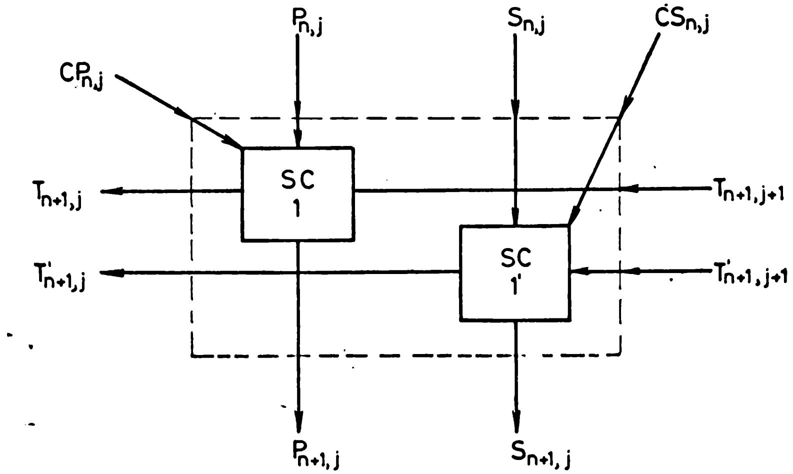


Fig.3-34. Celula din poziţiile 25...28 ale rîndului suplimentar.

In cadrul blocului 2 din fig.3-33 se realizează funcţia logică identitate :

$$ID_{n+1,j} = ID_{n+1,j-1}\left[(P_{n,j}LOG+S_{n,j}\overline{LOG})X_j+(\overline{P}_{n,j}LOG+\overline{S}_{n,j}LOG+\overline{S}_{n,j}\overline{LOG})\overline{X}_j\right]+ + ID_{n,j} \qquad (3\text{-}109)$$

In fig.3-35 se prezintă rîndul suplimentar de celule necesar pentru corecţia rezultatului şi pentru realizarea identităţii în cazul unor structuri logice celulare comasate cu anticiparea coeficienţilor de creştere, pentru $n=28$, $m\leqslant24$.

Ieşirile de transport $T_{29,1}$ şi $T'_{29,1}$ se utilizează pentru a se detecta un transport 1 spre poziţia 2^0 a antilogaritmului sau logaritmului cînd, aşa cum s-a mai arătat, trebuie prelucrat rezultatul în alt mod decît cel obişnuit.

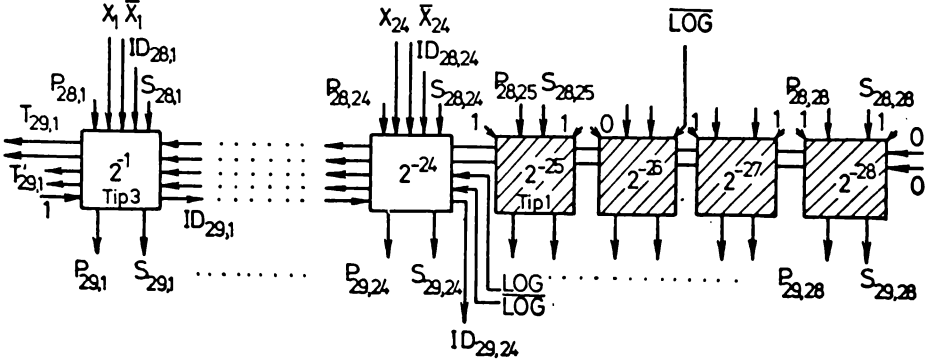


Fig.3-35. Rîndul suplimentar pentru realizarea identităţii şi corecţiei.

Intrarea $\overline{LOG}$ de la celula din poziția 2^{-26} realizează corecția cu 0 la logaritmare ($cor_2' = 1011$) și cu 1 la antilogaritmare ($cor_2'' = 1111$). La celelalte celule de tip 4 din ultimele poziții biții pentru corecție sînt aceiași la ambele operații (celulele s-ar putea deci simplifica).

Ieșirea $ID_{29,24}$ se poate utiliza la stabilirea momentului terminării descompunerii.

Prin utilizarea rîndului suplimentar în structura logică celulară durata maximă a descompunerii în factori sau în termeni crește relativ puțin.

Astfel, dacă se efectuează numai corecția pentru realizarea identității, aceasta poate duce la un transport cu lungime de propagare maximă de n celule după care începe o propagare în sens invers a funcției identitate peste m celule. Ținînd cont și de timpul de formare a sumei logice ($t_4=2\ t_p$) și a identității ($t_5=2t_p$) se poate considera că semnalul $ID_{n+1,m}$ - care indică terminarea descompunerii în structura asincronă - apare cu o întîrziere

$$t_{dmax} = (n+m+4)t_p = 2n\ t_p \qquad (3\text{-}110)$$

față de timpul $t_{df\ max}$ sau $t_{dt\ max}$ determinat în paragrafele anterioare. Rezultă deci durata operațiilor de descompunere pentru structura asincronă :

$$t_{dfasmax} = t_{dfmax} + t_{dmax} \qquad (3\text{-}111)$$

$$t_{dtasmax} = t_{dtmax} + t_{dmax} \qquad (3\text{-}112)$$

Pentru n=28, m=24, t_p= 6 ns, în cazul unor structuri logice comasate cu anticipare asincrone rezultă :

$$\underline{t_{dfasmax}} = t_{dfacmax} + t_{dmax} = 476\ t_p + 56\ t_p = \underline{532\ t_p = 3{,}192\ \mu s} \qquad (3\text{-}113)$$

și :

$$\underline{t_{dtasmax}} = t_{dtacmax} + t_{dmax} = 407\ t_p + 56\ t_p = \underline{463\ t_p = 2{,}778\ \mu s} \qquad (3\text{-}114)$$

Durata maximă a operațiilor de logaritmare și antilogaritmare se va stabili însă pe baza relațiilor (3-97) și (3-98) întrucît în alte cazuri decît cel mai defavorabil după apariția semnalului $ID_{n+1,m}$ mai pot să existe propagări ale transportului în ultimul rînd în cadrul operației de însumare a termenului L_n sau în cadrul operației de înmulțire cu factorul $(1+2^{-n})$.

Dacă se efectuează și corecția asupra logaritmului sau antilogaritmului, în cazurile pentru care durata descompunerii este maximă, nu apare un transport de lungime mare în rezultat și duratele maxime de mai sus ale descompunerilor în factori și termeni nu se modifică. Intrucît în unele cazuri, diferite de cel mai defavorabil, este posibil să intervină un transport de lungime maximă la adunarea corecţiei, după ce a apărut semnalul $ID_{n+1,m}$ de terminare a descompunerii, este necesar şi aici să se considere operaţiile de logaritmare şi antilogaritmare încheiate după un timp dat de relaţiile (3-97) şi (3-98) în care duratele $t_{dfasmax}$ şi $t_{dtasmax}$ sînt cele date de relaţiile (3-111) şi (3-112).

Starea iniţială a structurii logice celulare asincrone.

Referitor la utilizarea structurii logice celulare asincrone de logaritmare şi antilogaritmare trebuie arătat că starea iniţială a structurii (din momentul cînd se aplică la intrare operandul A sau L) poate avea o influenţă nefavorabilă asupra duratei operaţiei de descompunere în special în cazul duratelor mici.

Este deci necesar să se stabilească o stare iniţială de la care să se poată trece în timpul cel mai scurt la starea finală cauzată de operandul aplicat la intrare.

Ca stare iniţială s-a adoptat în calculele timpilor prezentate anterior şi la simularea structurilor logice celulare de descompunere - starea ce apare cînd la intrare este aplicat un operand nul. Dacă la intrarea structurilor de descompunere este aplicat un operand diferit de zero şi se aduce la un moment dat operandul egal cu zero are loc un proces tranzitoriu care în final conduce la starea "0" a structurii. In cazul cel mai defavorabil produsul final $P_n=0$ sau suma finală $S_n=0$ pe m biţi se stabilesc în această situaţie după un timp

$$t_{omax} = (n+m+1)t_p \qquad (3\text{-}115)$$

necesar pentru formarea şi propagarea pe orizontală a funcţiei ID peste m celule şi pentru propagarea pe verticală a biţilor lui P_i sau S_i peste n+1 rînduri prin intermediul circuitului SI-SAU condiţionat de funcţia $ID_{i,j}$.

In cazul cel mai favorabil se obţine timpul :

$$t_{omin} = (n+1)t_p \qquad (3\text{-}116)$$

cînd propagarea pe orizontală nu mai apare. Durata minimă corespunde existenţei iniţial la intrare a unui operand ce conţine un singur bit 1 în poziţia 2^{-m} iar durata maximă t_{omax} corespunde existenţei la intrare în momentul iniţial a unui operand ce conţine un singur bit 1 în poziţia 2^{-1}.

După acelaşi timp t_{omax}, în cazul cel mai defavorabil, se stabileşte şi semnalul de identitate a lui P_n cu A sau S_n cu L pe

BUPT

m biţi în cazul aplicării la intrare a numărului zero.

Pentru n=28 şi m=24, t_p=6 ns rezultă :

$$t_{omax} = 318 \text{ ns}, \quad t_{omin} = 174 \text{ ns} \qquad (3\text{-}117)$$

După acest timp, cuprins în diferite situaţii între cele două limite date de relaţia (3-117), semnalul $ID_{m+n,m}$ rămîne la valoarea 1. Dacă se consideră această stare ca stare iniţială şi se aplică în momentul "0" operandul la intrarea structurii logice de logaritmare şi antilogaritmare semnalul $ID_{n+1,m}$ va reveni la 0 după un timp cuprins tot între $t_{omin} \ldots t_{omax}$. Apoi semnalul de terminare a descompunerii în factori sau termeni $ID_{n+1,m}$=1 apare după un timp t_d (fig.3-36).

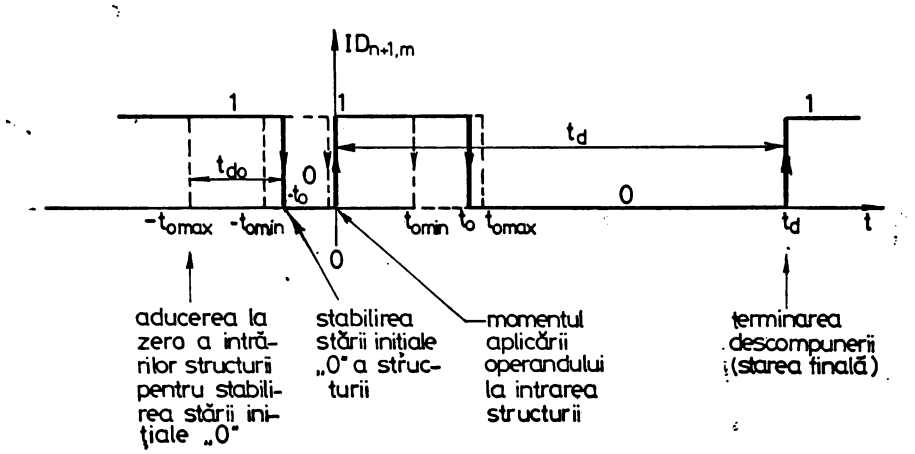


Fig.3-36. Variaţia semnalului de terminare a descompunerii.

Există însă cazuri în care timpul de descompunere este scurt, apropiat de t_{omin}. Astfel este posibil ca :

$$t_d = t_{omin} + 4\, t_p \qquad (3\text{-}118)$$

caz ce apare atunci cînd se descompune un operand ce conţine un singur bit 1 în poziţia 2^{-m}. Diferenţa 4 t_p se datoreşte timpului de formare a variabilei k_m (3 t_p) şi de formare a bitului produsului corect (t_p). Această diferenţă minimă se menţine între t_d şi t_o şi în cazurile cînd operandul este oarecare. In această situaţie semnalul $ID_{n+1,m}$ trece totuşi prin valoarea 0, aşa cum se arată în fig.3-36 şi se poate deci stabili momentul terminării descompunerii.

Numai în cazul descompunerii unui operand cu partea fracţionară nulă situaţia din structură rămîne nemodificată şi

BUPT

semnalul $ID_{n+1,m}$ nu mai ia valoarea 0 ci rămîne în continuare la valoarea 1 din starea inițială.

De aceea este necesar fie ca testarea sfîrșitului descompunerii să se facă numai după timpul t_{mmax} - aceasta constituind astfel cu aproximație durata minimă a operației de descompunere - fie ca operațiile cu operand nul (după virgulă) să se evite prin algoritmii dispozitivului aritmetic iar la testare să se țină cont de trecerea prin zero a semnalului $ID_{n+1,m}$ în intervalul t_d (adică de apariția unui semnal fals de terminare a descompunerii). In acest ultim caz durata minimă a operației de descompunere va fi aproximativ egală cu t_{omin}.

După apariția semnalului $ID_{n+1,m}$ corect, de terminare a descompunerii, mai trebuie, așa cum s-a arătat anterior, să se aștepte un timp $(n+2)t_p$ pentru ca rezultatul să fie corect în toate cazurile.

Dacă se evită prelucrarea unui operand nul atunci se poate realiza cu o structură logică celulară cu anticipare, comasare și asincronă o durată a operației de generare a logaritmului sau antilogaritmului cuprinsă între :

$$\left.\begin{array}{l} t_{log\ min} \\ t_{antilog\ min} \end{array}\right\} = (2\ n+7)t_p \qquad (3-119)$$

și

$$t_{log\ max} = t_{dfasmax} + (n+2)t_p \qquad (3-120)$$

$$t_{antilogmax} = t_{dtasmax} + (n+2)t_p \qquad (3-121)$$

Pentru n=28 și t_p= 6 ns rezultă :

$$\left.\begin{array}{l} \underline{t_{logmin}} \\ \underline{t_{antilogmin}} \end{array}\right\} = 63\ t_p = 378\ ns = \underline{0{,}378\ \mu s} \cong 0{,}4\ \mu s \qquad (3-122)$$

și

$$\underline{t_{log\ max}} = 562\ t_p = 3372\ ns = \underline{3{,}372\ \mu s} \cong 3{,}4\ \mu s \qquad (3-123)$$

$$\underline{t_{antilogmax}} = 493\ t_p = 2958\ ns = \underline{2{,}958\ \mu s} \cong 3\ \mu s \qquad (3-124)$$

BUPT

CAPITOLUL IV

SIMULAREA PE CALCULATOR NUMERIC A STRUCTURILOR LOGICE CELULARE DE LOGARITMARE ŞI ANTILOGARITMARE.

In scopul verificării calculelor efectuate în capitolul III al tezei şi a soluţiilor adoptate pentru structura logică celulară de logaritmare şi antilogaritmare s-a elaborat şi utilizat un program de simulare scris în FORTRAN /64/.

4.1. Program de simulare a unei scheme logice combinaţionale bazat pe urmărirea selectivă.

O schemă logică complexă, proiectată pe baza ecuaţiilor logice, trebuie să fie verificată, înainte de realizarea în forma finală, din punct de vedere funcţional, structural, al timpului de propagare, al încărcării elementelor, etc. [4,57]. Deoarece această verificare impune calculul funcţiilor logice ale ieşirilor schemei pentru un număr mare de combinaţii ale variabilelor de intrare este mai comod să se recurgă la simularea schemei logice pe calculator.

Simularea are în plus o serie de avantaje ca :

- semnalarea într-un timp foarte scurt a tuturor erorilor de proiectare ;
- verificarea schemei pentru toate combinaţiile posibile ale variabilelor de intrare ;
- furnizarea diagramelor de timp utilizate la controlul schemei fabricate ;
- furnizarea statisticilor privind încărcarea şi cantităţile elementelor din schemă ;
- posibilitatea reproiectării rapide, etc.

Chiar dacă toate variabilele de intrare ale unei scheme logice complexe se aplică în acelaşi moment, datorită timpului de propagare diferit după diferite direcţii, circuitele logice intermediare vor avea variabile de intrare ce iau valoarea finală - staţionară - în momente diferite. Deseori variabilele de intrare se modifică în timp sau ating valoarea staţionară după un "proces transitoriu" [57]. De aceea funcţia logică a unui circuit logic presintă în general o variaţie în timp ce depinde de modul de variaţie în timp al variabilelor de intrare. Modelarea unei scheme logice din punct de vedere al funcţionării în timp constituie problema esenţială la simulare [4, 57].

BUPT

Metoda cea mai des folosită şi cea mai avantajoasă pentru simularea funcţionării în timp a unei scheme logice o constituie "urmărirea selectivă". In cadrul acesteia se înregistrează şi se consideră numai acele momente în care variabilele de intrare ale schemei logice şi funcţiile logice se modifică. In afară de aceasta se mai utilizează metoda urmăririi la intervale de timp egale care este însă mai lentă iar datele ocupă un volum mult mai mare în memoria calculatorului.

In cele ce urmează se prezintă un program simplu de simulare a unei scheme logice bazat pe urmărirea selectivă. Acesta îşi propune ca scop principal determinarea modului de variaţie în timp al funcţiei logice şi nu se ocupă de descrierea topologică a schemelor logice sau de localizarea erorilor în schema logică /64/. Programul poate furniza diagrama de timp a unei funcţii logice pentru toate combinaţiile variabilelor de intrare. Pe baza acestera se poate verifica dacă schema logică este corect întocmită şi se poate stabili timpul de propagare al semnalelor logice prin schemă. Reproiectarea schemei logice duce la modificări simple ale programului astfel încît un număr mare de variante ale schemei poate fi verificat. In acest program descrierea schemelor logice se face prin ecuaţii.

Programul conceput, scris în FORTRAN pentru calculatorul FELIX C-256, se pretează în special la simularea structurilor logice celulare unde complexitatea schemei se datoreşte doar numărului mare de celule ce au de fapt scheme logice relativ simple. Programul de simulare este utilizat în acest caz pentru fiecare funcţie logică realizată de celulă şi este apoi repetat pentru fiecare celulă din structură.

In literatura de specialitate nu sînt prezentate programe concrete de simulare care să poată fi preluate de utilizatori.

Reprezentarea variaţiei în timp a variabilelor logice.

La baza simulării unor scheme logice prin metoda urmăririi selective stă diagrama de timp a variabilelor logice /64/. Aceasta constituie un tabel de forma Tabelului 4-1 avînd coloane pentru numărul de ordine, momentul modificării şi valoarea variabilei logice după modificare.

Tabelul are următoarele particularităţi importante :

- Momentul iniţial (din primul rînd al tabelului) trebuie să fie acelaşi pentru toate tabelele variabilelor de intrare ale schemei logice şi este considerat ca moment de referinţă. In cele ce urmează s-a ales ca moment iniţial $T_1 = 0$.

BUPT

Tabelul 4-1

Variabila V , NR..		
Număr de ordine	Momentul modificării	Valoarea logică
1	$T_1 = 0$	0
2	T_2	1
3	T_3	0
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
M_{MAX}	T_{MMAX}	V_{MMAX}

- Pentru timp se pot folosi orice fel de unități. In cazul cînd se utilizează în schema logică circuite integrate din aceeași familie se poate adopta ca unitate de timp chiar timpul mediu de propagare caracteristic. In coloana timpului momentele sînt trecute în ordine crescătoare.

- Numărul maxim de poziții din fiecare tabel este o mărime importantă în procesul de simulare și trebuie întotdeauna determinat prin însăși programul de simulare.

- In tabel apar numai momentele cînd variabila logică se modifică deci apar numai valori logice ce alternează.

Pe baza acestei proprietăți, pentru economie de volum de memorie în calculator, este suficient să se păstreze doar valoarea inițială (din rîndul 1). Pentru comoditate de scriere a programului este bine să se păstreze și cea de a doua valoare, astfel că datele incadrate în Tabelul 4-1 se pot elimina. Adresarea la valoarea corespunzătoare din tabel a funcției se face prin apelarea la rîndul 1 sau 2 după cum numărul de ordine corespunzător este impar sau par.

Toate variabilele de intrare ale unei scheme logice vor prezenta un astfel de tabel. In cazul în care variabila nu se modifică în timp tabelul conține un singur rînd.

Funcția logică de la ieșirea schemei va prezenta și ea un astfel de tabel. Problema principală ce apare la simularea schemei logice constă în întocmirea tabelului funcției logice pe baza următoarelor date : tabelele variabilelor de intrare, ecuația logică a schemei și timpii de propagare ai circuitelor logice componente. Tabelul obținut pentru funcția logică trebuie să îndeplinească deasemenea condițiile enumerate mai sus întrucît funcția logică obținută poate constitui o variabilă de intrare pentru alte scheme logice.

Deoarece negațiile unor variabile de intrare necesare la calculul funcției logice au în tabel momente diferite față de va-

BUPT

riabilele negate ele vor fi considerate şi tratate ca variabile de intrare independente.

Programul de simulare /64/.

Se prezintă mai jos un segment de program destinat simulării unei scheme logice simple, descrisă prin expresia logică. In cazul unor scheme mai complicate segmentul trebuie refolosit pentru fiecare nod în care interesează diagrama de timp a funcţiei logice respective.

Se consideră că în programul principal au fost stabilite anterior valorile variabilelor de intrare. Programul prezentat în fig.4-1 îndeplineşte următoarele sarcini :

- pe baza timpilor din tabela variabilelor de intrare stabileşte momentul în care urmează a se calcula funcţia logică (subrutina MOMENT) ;
- verifică dacă numărul de poziţii din tabelul funcţiei nu depăşeşte spaţiul rezervat în memorie (subrutina DEPASIRE);
- calculează funcţia logică pe baza expresiei logice cunoscute ;
- formează tabelul funcţiei prin eliminarea poziţiilor în care funcţia nu s-a modificat faţă de poziţia anterioară (subrutina LISTA).

Programul este parcurs ciclic pînă ce se epuizează toate poziţiile din tabelele variabilelor de intrare. In fig.4-1 blocurile au următoarele semnificaţii :

0. Programul principal ce include şi stabilirea tabelelor variabilelor de intrare.

1. Iniţializarea poziţiilor din tabelele variabilelor şi funcţiei şi precizarea poziţiilor finale din tabelele variabilelor.

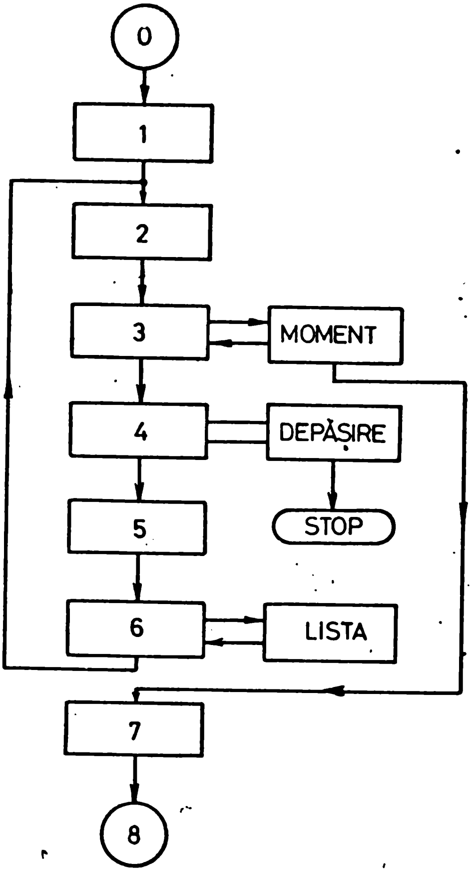


Fig.4-1 Segment din programul de simulare.

2. Stabilirea poziţiilor şi momentelor din tabelele variabilelor ce urmează a se compara.

3. Determinarea momentului de calcul al funcţiei logice şi a momentelor următoare din tabelele variabilelor, verificarea epuizării tabelelor (subrutina MOMENT).

4. Verificarea depăşirii spaţiului rezervat în memorie pentru tabelul funcţiei (subrutina DEPASIRE).

5. Calculul funcţiei logice pe baza expresiei logice.

6. Eliminarea poziţiei excedentare din tabelul funcţiei (subrutina LISTA).

7. Tipărirea diagramei de timp a funcţiei logice.

8. Programul principal ce cuprinde simularea altor scheme logice.

Pentru înţelegerea programului se prezintă mai jos notaţiile utilizate :

$L = 1 \ldots L_{MAX}$ - numerotarea variabilelor de intrare (L_{MAX} trebuie concretizat în program),

VL - variabilă cu numărul L,

M - numărul poziţiei din tabel în general ,

MVL- poziţia din tabelul variabilei VL,

T - timpul din tabel în general,

TVL(MVL) - timpul corespunzător poziţiei MVL din tabel,

TCF- timpul de calcul al funcţiei,

MREZ - numărul de poziţii rezervate pentru tabelul funcţiei logice,

MF - poziţie în tabelul funcţiei,

MF2 = 2 - modul de 2 al poziţiei MF (pentru a se stabili dacă MF este impară sau pară)

F(MF) - valoarea funcţiei logice din poziţia MF,

FN şi FV - valoarea nouă şi veche a funcţiei logice,

TFCT şi TF(MF) - momentul corespunzător poziţiei MF din tabelul funcţiei,

MMF - numărul maxim de poziţii din tabelul funcţiei,

NRF - numărul de ordine al funcţiei logice,

NRF_{MAX} - numărul de funcţii logice ce se simulează într-un program,

TINT - timpul de întîrziere (propagare) "unificat" al schemei logice.

Prin timp de întîrziere unificat se înţelege timpul de propagare a semnalelor logice între intrările şi ieşirea circuitului logic atunci cînd între fiecare intrare şi ieşire se reali-

BUPT

zează același timp de propagare (același număr de nivele logice). Dacă un circuit logic nu îndeplinește această condiție, pentru a se putea aplica programul propus, el trebuie descompus în mai multe circuite logice care îndeplinesc condiția de uniformitate a timpului de propagare între intrări și ieșire. De exemplu, pentru circuitul din fig.4-2, între intrarea V1 și ieșirea F sînt incluse trei nivele logice în timp ce între celelalte intrări și ieșire sînt incluse două nivele logice. Pentru a se putea aplica programul de simulare propus este necesar să se ia în considerare în locul variabilei V1 - variabila V1' pentru care se stabilește ușor diagrama de timp pe baza diagramei de timp a variabilei V1 și a timpului de propagare din ramura cuprinsă între V1 și V1'. In noua diagramă de timp (noul tabel) se va păstra însă nemodificat momentul de referință.

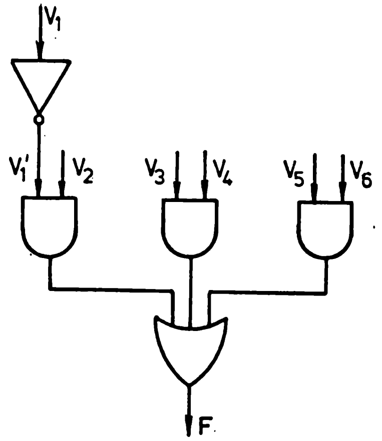


Fig.4-2 Circuit logic cu timp de întîrziere neunificat.

Se prezintă mai jos programul în FORTRAN corespunzător ordinogramei din fig.4-1.

```
Blocul 1 :    1 MV1 = 0
                MV2 = 0
                .
                .
                .
                MVL_MAX = 0
                MF = 0
                M MAX(1) = MMV1
                M MAX(2) = MMV2
                .
                .
                .
                M MAX(L_MAX) = MMVL_MAX
Blocul 2 :    2 T(1) = TV1 (MV1+1)
                T(2) = TV2 (MV2+1)
                .
                .
                .
                T(L_MAX) = TVL_MAX(MVL_MAX+1)
                M(1) = MV1+1
                M(2) = MV2+1
```

```
                  .
                  .
                  .
                  M(L_MAX) = MVL_MAX+1
Blocul 3 :        CALL MØMENT (T,M,MMAX,L_MAX,TCF, &6)
                  MV1  = M(1)-1
                  MV2  = M(2)-1
                  .
                  .
                  .
                  MVL_MAX = M(L_MAX)-1
Blocul 4 :        CALL DEPASIRE (MREZ,MF,NRF,  &7)
Blocul 5 :        MV12 = 2 - MØD (MV1, 2)
                  MV22 = 2 - MØD (MV2, 2)
                  .
                  .
                  .
                  MVL_MAX2 = 2- MØD(MVL_MAX,2)
                  MF2  = 2 - MØD (MF, 2)
                  F(MF)= Expresie logică
Blocul 6 :        IF(MF-1) 3,3,4
                3 FV = .NØT.F(1)
                  GØ TØ 5
                4 FV = F(3-MF2)
                5 FN = F(MF2)
                  CALL LISTA (FV,FN,TCF,TFCT,TINT,MF)
                  F(MF2) = .NØT.FV
                  MMF = MF
                  TF(MF) = TFCT
                  GØ TØ 2
Blocul 7 :      6 . . . . . . . . .
                7 . . . . . . . . .
```

In cadrul blocurilor 1 şi 2 se formează trei tablouri de variabile T,M,MMAX ce urmează a fi transmise subrubinei MOMENT. Prin intermediul ciclului ce cuprinde blocurile 2...6 se prelucrează toate rîndurile din tabelele variabilelor şi se alcătuieşte tabelul funcţiei logice. Programul se încheie atunci cînd tabelele variabilelor au fost epuizate, ieşirea din program făcîndu-se la blocul 7 prin subrutina MOMENT sau prin subrutina DEPASIRE (cînd se depăşeşte spaţiul rezervat în memorie pentru tabelul funcţiei).

In fig.4-3 se prezintă ordinegrama subrutinei MOMENT. L este numărul de ordine al variabilei de intrare. EG(L) reprezintă

BUPT

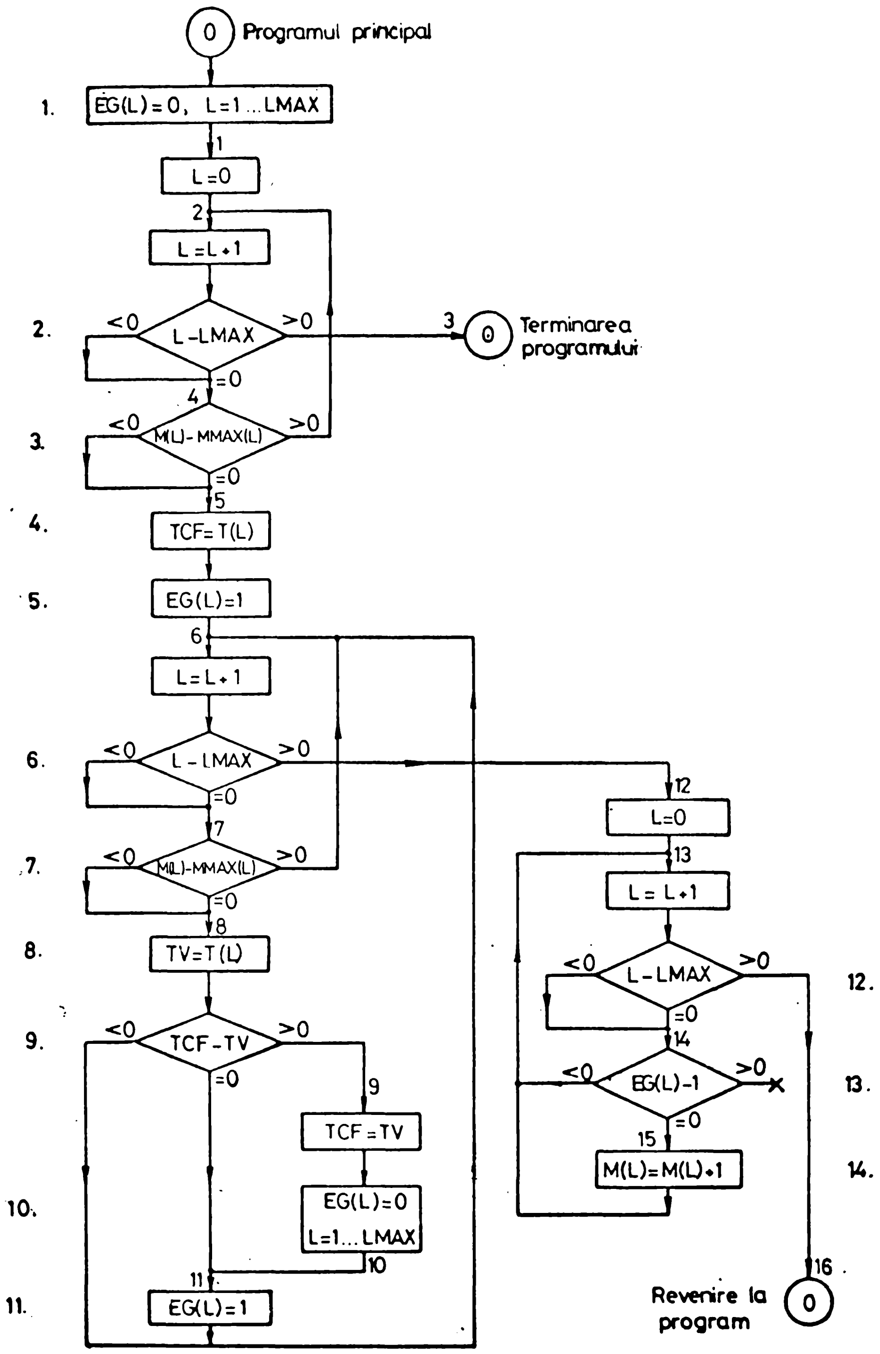


Fig.4-3 Ordinograma subrutinei MOMENT.

BUPT

indicatorul de egalitate a timpului din tabelul variabilei cu numărul L cu timpul cel mai mic dintre cele L_{MAX} momente ce se compară. In caz de egalitate EG(L)=1 iar în caz contrar EG(L)=0.

Blocurile din fig.4-3 au următoarea semnificaţie :

1. Anularea indicatorilor de egalitate.

2. Verificarea epuizării variabilelor.

3. Verificarea epuizării tabelelor variabilelor.

4. Stabilirea primului tabel neepuizat şi a timpului minim cu care se începe comparaţia (TCF).

5. Stabilirea primului indicator de egalitate.

6. Verificarea epuizării variabilelor.

7. Verificarea epuizării tabelului variabilei ce urmează după prima variabilă cu tabelul neepuizat.

8. Stabilirea momentului ce se compară cu TCF.

9. Comparaţia între două momente din două tabele pentru stabilirea timpului minim TCF.

10. Anularea indicatorilor de egalitate.

11. Stabilirea indicatorului de egalitate al variabilei L căreia îi corespunde TCF.

12. Verificarea epuizării variabilelor.

13. Cercetarea indicatorului de egalitate.

14. Trecerea la momentul următor din tabel dacă indicatorul de egalitate este egal cu 1.

In fig.4-3 sînt marcate de asemenea pe liniile de trecere de la un bloc la altul etichetele din programul FORTRAN corespunzător subrutinei.

Subrutina DEPASIRE este foarte simplă. In cadrul ei se măreşte cu 1 numărul de ordine al poziţiei din tabelul funcţiei, se întrerupe sau se trece la un alt segment şi se tipăreşte un avertisment dacă se depăşeşte spaţiul rezervat tabelului funcţiei în memorie.

In fig.4-4 se prezintă ordinograma subrutinei LISTA.

Blocurile din fig.4.4 au următoarea semnificaţie :

1. Verificarea numărului de poziţii din tabelul funcţiei completate pînă în acel moment.

2. Compararea valorii calculate a funcţiei logice cu valoarea din poziţia anterioară dacă numărul de poziţii din tabel este > 1.

3. Pentru primul rînd din tabel se egalează timpul funcţiei cu TCF care în acest caz este egal cu 0 şi se păstrează momentul de referinţă.

BUPT

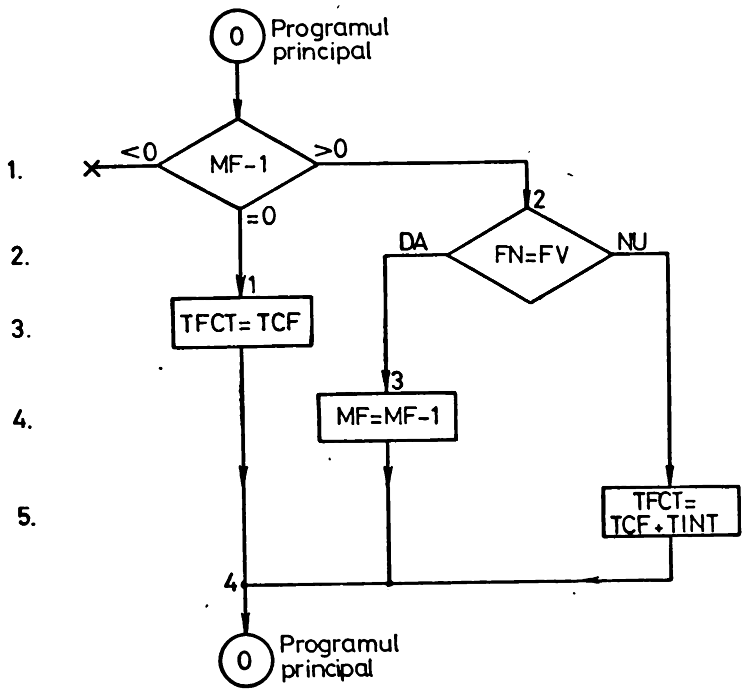


Fig.4-4. Ordinograma subrutinei LISTA .

4. Anularea din tabelul funcţiei a poziţiei în care funcţia nu s-a modificat.

5. Adăugarea timpului de întîrziere unificat.

In cazul cînd se operează cu primul rînd din tabelul funcţiei momentul iniţial trebuie să fie zero deci acelaşi cu momentul iniţial din tabelele variabilelor. De aceea în fig.4-4, în ramura 1-4 a subrutinei nu se mai adaugă timpul de propagare prin schema logică.

Valoarea iniţială a funcţiei logice va fi dată aici de valorile iniţiale ale variabilelor de intrare.

In continuare se prezintă cele trei subrutine scrise în FORTRAN. In cadrul subrutinei MOMENT declaraţiile de tip pentru tablouri s-au scris pentru $L_{MAX}=6$ dar de fapt numărul variabilelor de intrare nu este limitat.

```
      SUBRØUTINE MØMENT (T,M,MMAX,LMAX,TCF, *)
      DIMENSIØN M(6),MMAX(6)
      INTEGER T(6),TCF, TV
      DØ 1 I=1, LMAX
      EG(I) = 0
    1 CØNTINUE
      L=0
    2 L=L+1
      IF(L-LMAX)4,4,3
    3 RETURN 1
    4 IF(M(L)-MMAX(L))5,5,2
    5 TCF=T(L)
      EG(L)=1
    6 L=L+1
      IF(L-LMAX)7,7,12
    7 IF(M(L)-MMAX(L))8,8,6
    8 TV=T(L)
      IF(TCF-TV)6,11,9
    9 TCF=TV
      DØ 10 I=1, LMAX
      EG(I)=0
   10 CØNTINUE
   11 EG(L)=1
      GØ TØ 6
   12 L=0
   13 L=L+1
      IF(L-LMAX)14,14,16
   14 IF(EG(L)-1)13,15,15
   15 M(L)=M(L)+1
      GØ TØ 13
   16 RETURN
      END

      SUBRØUTINE LISTA (FV,FN,TCF,TFCT,TINT,MF)
      LØGICAL FV,FN
      INTEGER TFCT,TINT,TCF
      IF(MF-1)1,1,2
    1 TFCT=TCF
      GØ TØ 4
    2 IF(FN.AND.FV.ØR..NØT.FN.AND..NØT.FV)GØ TØ 3
      TFCT=TCF+TINT
      GØ TØ 4
```

BUPT

```
3   MF=MF-1
4   RETURN
    END

    SUBRØUTINE DEPASIRE (MREZ,MF,NRF, *)
    DIMENSIØN MREZ (NRFMAX)
    MF=MF+1
    IF(MREZ(NRF)-MF)1,3,3
1   WRITE (108,2) NRF
2   FØRMAT (' MREZ DEPASIT LA FUNCTIA NR',2X,I2)
    RETURN 1
3   RETURN
    END
```

Segmentul de program prezentat mai sus şi subrutinele aferente au fost verificate şi utilizate în cadrul programelor de simulare a structurilor logice celulare de logaritmare şi antilogaritmare. Cu ajutorul acestora s-a simulat uşor schema complexă constituită de o structură logică celulară pentru numere cu 28 de biţi.

4.2. Simularea structurilor logice celulare de generare a logaritmilor şi antilogaritmilor.

Pentru verificarea rezultatelor obţinute în Capitolele II şi III, la studiul algoritmului bazat pe descompunerea în factori şi termeni şi la studiul structurilor logice celulare de generare a logaritmilor şi antilogaritmilor s-a utilizat simularea pe calculatorul numeric FELIX C-256 a structurilor.

Intrucît generarea antilogaritmilor este asemănătoare cu generarea logaritmilor s-au stimulat numai structurile logice pentru generarea logaritmilor, concluziile rămînînd valabile şi pentru cealaltă operaţie.

Au fost simulate astfel următoarele structuri :

- structura logică celulară de descompunere în factori simplă (fig.3-7) realizată cu celulele de forma dată în fig.3-8,
- structura logică celulară de însumare a termenilor (fig.3-10) realizată cu celula din fig.3-11,
- structura logică celulară de descompunere în factori cu anticipare (fig.3-25) realizată cu celula din fig.3-22.

La alcătuirea celulelor s-au considerat circuite logice integrate de tip TTL rapide [69] avînd toate acelaşi timp de propagare pe un nivel logic t_p. De aceea timpul t_p s-a adoptat drept unitate. Acest considerent a fost impus de faptul că în cazul adoptării unor timpi de propagare diferiţi pentru diversele circuite logice dintr-o celulă ar fi rezultat tabele ale variabilelor şi funcţiilor (Tabelul 4-1) cu un număr foarte mare de rînduri şi acestea nu ar mai fi încăput în memoria internă a calculatorului. Chiar şi dacă s-a considerat acelaşi timp t_p pentru toate circuitele logice programele de simulare (împreună cu subrutinele ataşate de editorul de legături) au ocupat un volum de memorie internă de cca. 64 koctet. Secţionarea programului de simulare, care ar fi permis micşorarea volumului de memorie internă necesar la un moment dat pentru calcule prin folosirea unei memorii externe lente de capacitate mare, nu a fost practic posibilă din cauza naturii ciclice a programului şi a necesităţii permanente a schimbului de informaţie cu memoria externă ceea ce ar fi condus la o creştere exagerată a duratei simulării în timp ce rezultatele nu ar fi diferit prea mult faţă de cele obţinute în ipoteza făcută.

In acest fel simularea unei descompuneri în factori sau a unei însumări de termeni pentru un operand aplicat la intrările unei structuri logice celulare cu 28 de biţi durează 1...2 minute iar compilarea şi editarea legăturilor programului - cca 2 minute. Acest timp relativ redus a permis simularea unui număr mai mare de exemple.

Programele de simulare scrise în FORTRAN, inclusiv segmentele pentru tipărirea rezultatelor şi subrutinele aferente includ 400-600 cartele.

Cu ajutorul programelor de simulare s-au cercetat următoarele aspecte ale funcţionării structurilor logice celulare de descompunere în factori şi de însumare a termenilor :

- durata maximă a descompunerii în factori şi a generării logaritmului,
- traseul de propagare cu lungime maximă a transportului şi împrumutului,
- reducerea duratei maxime a descompunerii prin modificarea modului de realizare concretă a celulelor,
- delimitarea regiunii active a structurii logice de descompunere în factori cu anticipare,
- timpul de stabilire a stării iniţiale "0".

Rezultatele obţinute prin simularea unor structuri logice celulare pentru numere cu 28 biţi au coincis în foarte bună măsură cu cele deduse teoretic în Capitolele II şi III, atît în ceea ce priveşte erorile la generarea logaritmului cît şi durata maximă a generării acestuia.

BUPT

Simulîndu-se o serie de exemple, alese astfel încît durata descompunerii să fie cît mai mare, s-au confirmat ipotezele privind cazurile cele mai defavorabile din punct de vedere al duratei, admise la calculul analitic al timpului în Capitolul III. Astfel, traseul de propagare cu lungime maximă a transportului şi împrumutului rezultat la simulare a coincis cu cel presupus în Capitolul III.

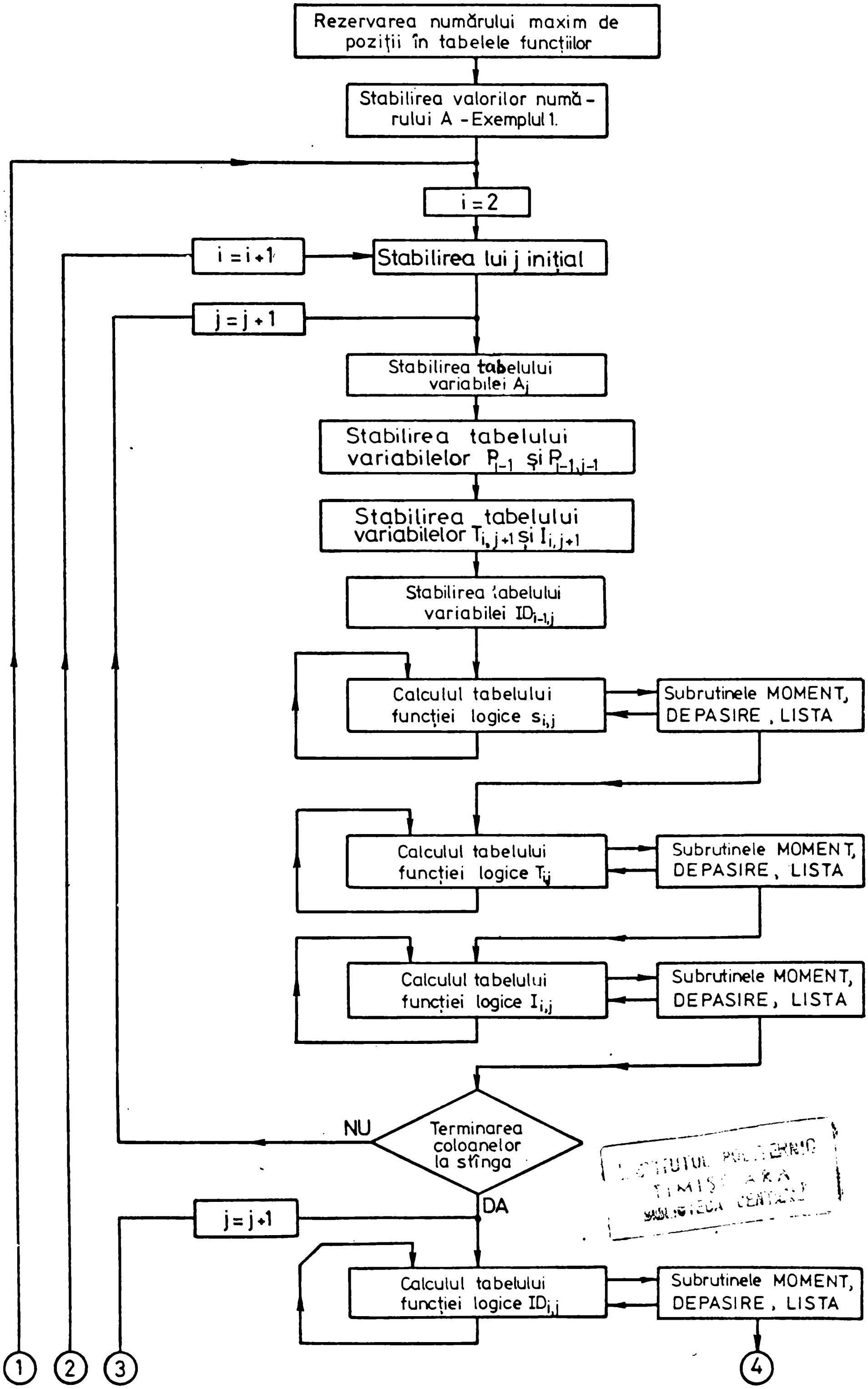
Prin încercarea mai multor variante de realizare a celulelor structurilor, folosind circuite integrate TTL, s-au găsit celule care asigură o durată minimă a descompunerii în factori. Este vorba de celulele prezentate în fig.3-8, 3-1o, 3-22 şi 3-23 în care suma este realizată cu două nivele logice de circuite NU-SI (negaţiile necesare la intrările acestora se obţin cu invertoare) iar transportul este realizat prin circuite SI-SAU.

In fig.4-5 se prezintă ordinograma programului de simulare a structurii logice celulare de descompunere în factori cu anticipare (fig.3-25) în care notaţiile corespund celor utilizate la scrierea ecuaţiilor logice (3-72)...(3-77).

Ca anexă la teză se prezintă rezultatul obţinut la simularea structurii logice celulare de descompunere în factori cu anticipare pentru cazul operandului A=1,11...11 care la descompunerea fără anticipare conduce la cazul cel mai defavorabil din punct de vedere al duratei (Capitolul III). Se poate remarca faptul că pentru n=28 în locul unei durate de 53o t_p (relaţia 3-38) rezultă prin anticipare o durată de~14o t_p (timpul de stabilire a produsului în ultimul rînd al structurii).

Pe anexă este marcată cu verde propagarea transportului în interiorul zonei de identitate (care este delimitată în partea dreaptă cu linie albastră). Această propagare a transportului nu influenţează asupra duratei totale a descompunerii în factori. Traseul de întîrziere care impune aici durata descompunerii în factori este marcat cu linie roşie în anexă. Se remarcă deci efectul anticipării în structura logică celulară de descompunere în factori.

BUPT


Rezervarea numărului maxim de poziții în tabelele funcțiilor
Stabilirea valorilor numărului A - Exemplul 1.
i = 2
i = i + 1
Stabilirea lui j inițial
j = j + 1
Stabilirea tabelului variabilei A_j
Stabilirea tabelului variabilelor P_{i-1} și $P_{i-1,j-1}$
Stabilirea tabelului variabilelor $T_{i,j+1}$ și $I_{i,j+1}$
Stabilirea tabelului variabilei $ID_{i-1,j}$
Calculul tabelului funcției logice $s_{i,j}$
Subrutinele MOMENT, DEPASIRE, LISTA
Calculul tabelului funcției logice $T_{i,j}$
Subrutinele MOMENT, DEPASIRE, LISTA
Calculul tabelului funcției logice $I_{i,j}$
Subrutinele MOMENT, DEPASIRE, LISTA
Terminarea coloanelor la stînga
NU
DA
j = j + 1
Calculul tabelului funcției logice $ID_{i,j}$
Subrutinele MOMENT, DEPASIRE, LISTA
1
2
3
4

BUPT

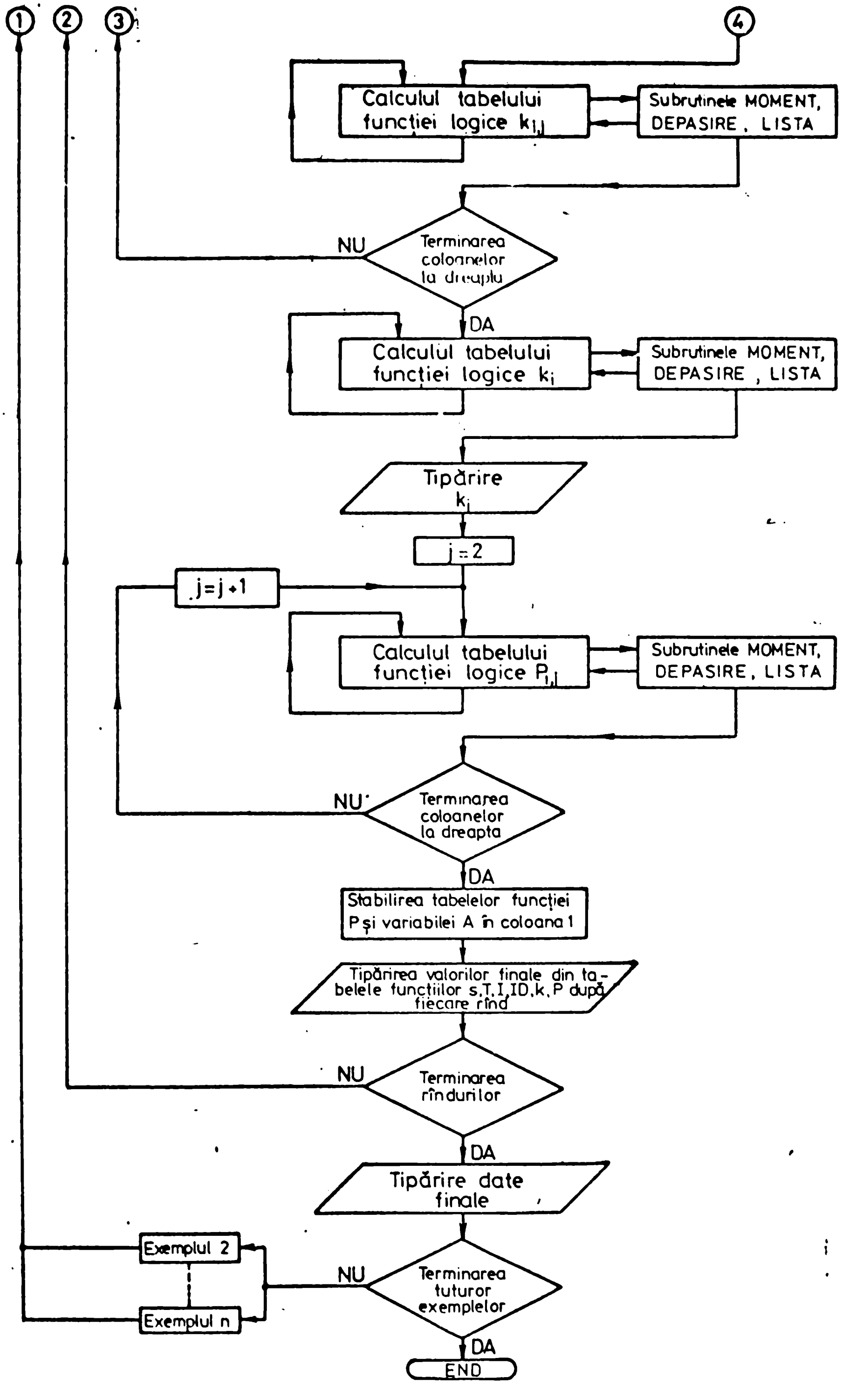


Fig.4-5 Ordinograma programului de simulare a structurii logice celulare de descompunere în factori cu anticipare.

CAPITOLUL V

DISPOZITIV ARITMETIC CU VIRGULA MOBILA LOGARITMIC

In Capitolul III s-a arătat posibilitatea generării logaritmilor şi antilogaritmilor binari cu ajutorul unei structuri logice celulare. In cazul unei structuri cu anticipare, cu comasare şi asincronă rezultă duratele minime şi maxime (relaţiile 3-122, 3-123, 3-124) :

$$t_{log\ min} = t_{antilog\ min} = 63\ t_p \cong 0,4\ \mu s \qquad (5-1)$$

$$t_{log\ max} = 562\ t_p \cong 3,4\ \mu s \qquad (5-2)$$

$$t_{antilog\ max} = 493\ t_p \cong 3\ \mu s \qquad (5-3)$$

dacă n = 28 iar circuitele integrate utilizate au t_p = 6 ns.

Considerînd că durata medie a generării logaritmului şi antilogaritmului este media aritmetică a duratelor limită de mai sus rezultă :

$$t_{log\ med} = 1,9\ \mu s \qquad (5-4)$$

$$t_{antilog\ med} = 1,7\ \mu s \qquad (5-5)$$

Ţinînd cont că durata operaţiilor de înmulţire şi împărţire într-un dispozitiv aritmetic paralel cu virgulă mobilă obişnuit realizat cu acelaşi tip de circuite integrate [74] este de ordinul 24...29 μs, este avantajos, din punct de vedere al vitezei de calcul, să se realizeze un dispozitiv aritmetic paralel cu virgulă mobilă logaritmic. Acest dispozitiv va realiza adunarea şi scăderea după algoritmii obişnuiţi [55] iar înmulţirea şi împărţirea cu ajutorul logaritmilor. Fără o complicare prea mare se poate prevedea în plus la acest dispozitiv realizarea operaţiilor de ridicare la patrat extragerea rădăcinii patrate, logaritmarea şi antilogaritmarea (acestea din urmă în scopul calculării puterilor şi radicalilor de orice ordin precum şi a logaritmilor şi antilogaritmilor în orice bază) /71/. Prin urmare dispozitivul aritmetic logaritmic poate efectua în total 8 operaţii spre deosebire de un dispozitiv aritmetic obişnuit. Dacă dispozitivul este utilizat în calcule ştiinţifice - inginereşti el permite rezolvarea unor programe fără a mai apela la o serie de subrutine pentru unele calcule matematice, rezultînd o economie suplimentară de timp de calcul.

Pentru a se asigura aceeaşi precizie în operaţiile cu logaritmi ca şi la un dispozitiv obişnuit cu lungime simplă de cuvînt, structura logică celulară se generează logaritmul şi antilogaritmul

BUPT

trebuie să conţină circuite pentru un număr mai mare de biţi. Astfel, pentru operanzi cu 24 biţi după virgulă este necesară o structură cu 28 de rînduri şi coloane. Eroarea rezultatului este provocată de eroarea de la generarea logaritmului şi antilogaritmului. Pentru o structură logică celulară cu 28 biţi eroarea maximă posibilă este de ordinul 2^{-24} (Capitolul II). Ea se poate reduce la 2^{-25} dacă structura se realizează cu 29 biţi sau printr-o corecţie, aşa cum s-a arătat în Capitolul II.

In ceea ce priveşte reprezentarea numerelor în virgulă mobilă în calculator, cazul cel mai favorabil din punct de vedere al vitezei dispozitivului aritmetic logaritmic este acela al reprezentării binare semn-mărime cînd numerele sînt normalizate în domeniul $[1, 2)$. In acest caz se evită o serie de microoperaţii legate de aducerea numărului în domeniul impus de algoritm.

Este posibil să se aplice algoritmii pentru operaţii prezentaţi mai jos şi în cazul reprezentării hexadecimale [76] fiind însă necesară transformarea în binar a exponentului şi aducerea părţii fracţionare (normalizată în hexadecimal) în domeniul $[1, 2)$. Aceste transformări se fac prin simple deplasări. Reprezentarea hexadecimală este avantajoasă în dispozitivul aritmetic cu virgulă mobilă logaritmic datorită reducerii mari a duratei normalizării dar lungeşte operaţiile fără normalizare şi reduce precizia părţii fracţionare.

In cazul unor operanzi reprezentaţi în complement de 2 este necesară recomplementarea pentru reprezentarea în semn-mărime întrucît nu există relaţii simple între logaritmul unui număr şi logaritmul complementului de 2 al numărului.

In cele ce urmează se vor prezenta mai întîi algoritmii elaboraţi pentru cele 6 operaţii aritmetice cu logaritmi, considerîndu-se operanzii reprezentaţi în binar, în semn-mărime, normalizaţi în modul obişnuit, adică în domeniul $[0, 1)$ şi avînd exponentul polarizat /71/.

Algoritmul pentru adunare şi scădere nu utilizează logaritmii şi poate fi deci algoritmul cunoscut pentru dispozitive aritmetice cu virgulă mobilă [55]. De aceea el nu s-a mai prezentat aici.

5.1. Algoritmi pentru operaţiile cu logaritmi /71/

Pentru elaborarea amănunţită a algoritmilor operaţiilor aritmetice s-a adoptat formatul operanzilor din fig.5-1 care este identic cu cel folosit în "operatorul cu virgulă flotantă" al calculatoarelor FELIX C-256 şi IRIS-50 [75,76] pentru lucrul cu lungime

BUPT

simplă de cuvînt.

Cele două părți ale numărului fracționar au fost denumite în fig.5-1 "exponent polarizat" (EP) şi "parte fracţionară" (PF) pentru ca noţiunile de "caracteristică" şi "mantisă" să poată fi utilizate la exprimarea logaritmilor, evitîndu-se astfel orice confuzie. Exponentul este polarizat cu 64 fiind întotdeauna pozitiv. S reprezintă semnul părţii fracţionare.

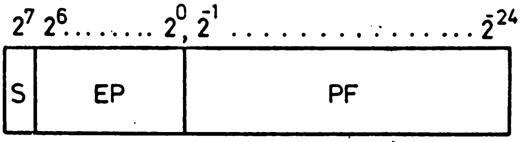


Fig.5-1. Formatul operanzilor.

Operaţiile cu doi operanzi se efectuează în forma :

$$\text{Operandul I} \left\{ \begin{matrix} + \\ - \\ \times \\ : \end{matrix} \right\} \text{operandul II}$$

operandul I fiind într-un registru general [76] iar operandul II în memorie sau într-un alt registru general.

In cazul cînd rezultă un exponent polarizat negativ (subdepăşirea exponentului) sau parte fracţionară nulă se prevede egalarea cu zero a numărului. Numărul zero are partea fracţionară şi exponentul polarizat nule.

5.1.1. Algoritmul pentru înmulţire şi împărţire

In cadrul dispozitivului cu virgulă mobilă logaritmic aceste două operaţii sînt practic identice. Ele necesită generarea logaritmilor celor doi operanzi, adunarea sau scăderea logaritmilor şi generarea antilogaritmului rezultatului adunării sau scăderii. Etapele ce trebuie parcurse în ciclul de execuţie al instrucţiei sînt următoarele :

a) Se preia primul operand dintr-un registru general al calculatorului.

b) Dacă operandul este nul se stabileşte direct rezultatul nul. Dacă operandul nu este nul se memorează semnul lui şi în locul lui în registru se înscrie 0.

c_1) Se trimit ultimii 23 de biţi ai părţii fracţionare a primului operand la intrările structurii logice celulare pentru generarea logaritmului. Intrucît virgula s-a considerat deplasată cu o poziţie la dreapta pentru aducerea părţii fracţionare în do-

BUPT

meniul 1, 2), va fi necesară corecţia cu -1 a exponentului polarizat (corecţia se efectuează la punctul g_2).

De la structura logică celulară se obţine în mod asincron mantisa logaritmului părţii fracţionare a operandului cu 24 de biţi care se înscriu în poziţiile $2^{-1}...2^{-24}$ ale registrului operandului (fig.5-2). Caracteristica logaritmului este chiar exponentul operandului deci rezultă o caracteristică polarizată.

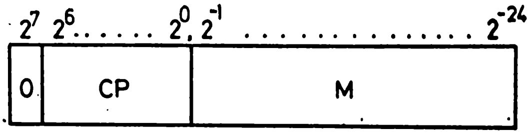


Fig.5-2. Formatul logaritmului.

In fig.5-2, cu CP s-a notat caracteristica polarizată iar cu M mantisa logaritmului binar.

c_2) In paralel cu generarea logaritmului operandului I se citeşte din memorie sau dintr-un registru general al calculatorului cel de al doilea operand.

d) Dacă operandul II este nul, în cazul înmulţirii se stabileşte un rezultat nul iar în cazul împărţirii se efectuează derutarea pentru împărţire cu zero. In caz contrar se memorează semnul operandului al doilea şi în locul lui în registru se înscrie zero.

e) Se trimit ultimii 23 de biţi ai părţii fracţionare a celui de al doilea operand la intrările structurii logice pentru generarea logaritmului. Corecţia cu -1 a exponentului polarizat, necesară datorită mutării virgulei cu o poziţie la dreapta pentru încadrarea părţii fracţionare a numărului în domeniul 1, 2), se face la punctul g_2.

De la structura logică celulară se obţine în mod asincron mantisa logaritmului cu 24 biţi care se introduce în poziţiile $2^{-1}...2^{-24}$ ale registrului celui de al doilea operand. Caracteristica polarizată a logaritmului este chiar exponentul polarizat.

f) In cazul înmulţirii se face adunarea celor doi logaritmi. In cazul împărţirii se face scăderea celor doi logaritmi prin adunarea complementului de 2 al logaritmului operandului al doilea inclusiv poziţia 2^7 pentru semn.

g_1) Mantisa rezultatului, cu 24 biţi, se trimite la structura logică celulară pentru generarea antilogaritmului. Rezultatul reprezentînd partea fracţionară a unui număr cuprins în domeniul 1, 2), se obţine în mod asincron cu 23 biţi care se plasează

BUPT

în pozițiile $2^{-2}...2^{-24}$ ale registrului pentru rezultat. In poziția 2^{-1} se înscrie 1. Virgula antilogaritmului, aflată după acest 1, trebuie mutată cu o poziție spre stînga pentru aducerea părții fracționare a rezultatului în domeniul [o,1). Corecția cu +1 a exponentului polarizat se face la punctul g_2.

Caracteristica polarizată a logaritmului rezultat la punctul f devine exponentul polarizat necorectat al rezultatului înmulțirii sau împărțirii. Rezultatul nu mai trebuie normalizat.

g_2) Se face corecția exponentului polarizat necesară din cauza mutărilor virgulei operanzilor la logaritmare și după antilogaritmare precum și din cauza adunării sau scăderii a două caracteristici ambele polarizate cu 64. Astfel la înmulțire se va aduna corecția 1011111 iar la împărțire 0100001 (complementul de 2 al corecției de la înmulțire).

h) Se verifică apariția unei supradepășiri sau a unei subdepășiri a exponentului polarizat al rezultatului. La supradepășire în pozițiile 2^7 și 2^6 ale exponentului polarizat apare perechea de biți 10 iar la subdepășire - perechea 11. La supradepășire se efectuează o derutare iar la subdepășire se anulează rezultatul.

i) Se stabilește semnul rezultatului și se introduce în poziția 2^7 a registrului rezultatului.

j) Se transferă rezultatul într-un registru general al calculatorului.

In cazul înmulțirii și împărțirii nu apar deplasări, nu se pierd prin aceasta cifre ale rezultatului și nu este necesară rotunjirea. Deasemenea, la împărțire nu este necesar ca deîmpărțitul să fie mai mic decît împărțitorul, ceea ce simplifică procedura.

5.1.2. Algoritmul pentru ridicare la patrat și extragerea rădăcinii patrate.

Aceste operații se realizează în mod asemănător într-un dispozitiv aritmetic cu virgulă mobilă logaritmic. Ele necesită generarea logaritmului operandului, deplasarea lui cu o poziție la stînga sau la dreapta și generarea antilogaritmului lui. Etapele ce trebuie parcurse în ciclul de execuție al instrucției sînt următoarele :

a) Se preia operandul dintr-un registru general sau din memoria calculatorului.

b) In cazul extragerii rădăcinii patrate se verifică dacă operandul este negativ cînd se efectuează o derutare. In cazul ridicării la patrat în poziția pentru semn se înscrie zero întrucît

BUPT

semnul nu contează. Se verifică dacă operandul este nul cînd se stabileşte direct rezultatul nul.

c) Se trimit ultimii 23 biţi ai părţii fracţionare a operandului la intrările structurii logice celulare pentru generarea logaritmului.

De la structura logică celulară se obţine în mod asincron mantisa logaritmului părţii fracţionare a operandului cu 24 de biţi care se introduc în poziţiile $2^{-1}...2^{-24}$ ale registrului operandului. Caracteristica polarizată a logaritmului este chiar exponentul polarizat.

Pentru extragerea rădăcinii patrate corecţia caracteristicii datorită mutării virgulei la generarea antilogaritmului şi pentru polarizare corectă se face înainte de deplasare, prin adunarea cantităţii 01000001 ca şi la împărţire.

d) Se efectuează deplasarea cu o poziţie la stînga, pentru ridicare la patrat sau la dreapta, pentru extragerea rădăcinii patrate, a logaritmului complet (caracteristica şi mantisa).

e) Pentru ridicarea la putere se face corecţia caracteristicii polarizate necesară datorită mutării virgulei la generarea logaritmului şi antilogaritmului şi pentru polarizare corectă. Se ţine cont că s-a efectuat o deplasare astfel că se adună corecţia 10111111, ca şi la înmulţire.

In cazul extragerii rădăcinii patrate se poate efectua o rotunjire a mantisei cînd prin deplasare la dreapta se pierde un bit 1 (1 în poziţia 2^{-25} după deplasare).

f_1) cei 24 de biţi ai mantisei logaritmului se trimit la intrările structurii logice celulare pentru generarea antilogaritmului. Rezultatul, reprezentînd partea fracţionară a unui număr cuprins în domeniul [1, 2) se obţine în mod asincron, cu 23 de biţi ce se plasează în poziţiile $2^{-2}...2^{-24}$ ale registrului rezultatului. In poziţia 2^{-1} se înscrie un 1.

Caracteristica polarizată devine exponent polarizat. Corecţia cu +1 a exponentului polarizat datorită mutării virgulei s-a efectuat la punctul c sau e. Rezultatul nu necesită normalizare.

f_2) Se verifică apariţia unei supradepăşiri sau a unei subdepăşiri a exponentului polarizat al rezultatului ca la înmulţire. La supradepăşire apare perechea de biţi 10 iar la subdepăşire - perechea 11 în poziţiile 2^7 şi 2^6 ale exponentului polarizat. Se efectuează o derutare în cazul supradepăşirii sau se anulează rezultatul în cazul subdepăşirii.

BUPT

g) Se transferă resultatul într-un registru general al calculatorului.

5.1.3. Algoritmul pentru logaritmare.

Această operaţie este necesară ca operaţie de sinestătătoare la efectuarea simplă, prin program sau eventual cablat, a unor operaţii mai complexe : ridicarea la o putere oarecare, extragerea rădăcinii de ordin oarecare calculul exponenţialelor şi a logaritmilor în altă bază. Pentru aceasta este obligatoriu ca logaritmul odată generat să fie reprezentat în virgulă mobilă ca orice operand. Reprezentarea aceasta se obţine prin normalizare obişnuită cu numărarea deplasărilor efectuate.

Operaţia se aplică numai numerelor pozitive şi diferite de zero. Logaritmul obţinut va fi pozitiv sau negativ după cum exponentul operandului, după corecţia cu -1 pentru mutarea virgulei, este pozitiv sau negativ.

Succesiunea etapelor din ciclul de execuţie al instrucţiei este următoarea :

a) Se preia operandul dintr-un registru general sau din memoria calculatorului.

b) Se verifică semnul operandului. Dacă acesta este negativ se efectuează o derutare.

c_1) Se trimit ultimii 23 de biţi ai părţii fracţionare a operandului la intrările structurii logice celulare pentru generarea logaritmului. Mantisa logaritmului, obţinută în mod asincron, cu 24 de biţi se plasează în poziţiile $2^{-1}...2^{-24}$ ale registrului pentru logaritm. Exponentul polarizat devine caracteristica polarizată.

c_2) Se face corecţia cu -1 a caracteristicii polarizate, pentru a se ţine cont de mutarea virgulei cu o poziţie la dreapta la generarea logaritmului. Scăderea acestui 1 se face prin adunarea complementului de 2, adică în binar : 11111111.

d) Dacă caracteristica este negativă se efectuează complementul de 2 al întregului operand inclusiv poziţia pentru semn. Prin aceasta se inversează semnul mantisei (se adună ± 1 la mantisă) şi se înlătură polarizarea caracteristicii.

e) Se înscrie 0 în poziţia 2^7 şi 2^6 a registrului logaritmului pentru ca eventualii biţi 1 să nu fie deplasaţi.

f) Dacă conţinutul poziţiilor $2^5...2^0$ este diferit de zero se face o normalizare la dreapta. Deplasările se numără într-un numărător (maximum 6 deplasări). Dacă conţinutul acestor poziţii este nul atunci se face eventual o normalizare la stînga. Deplasă-

BUPT

rile se numără într-un numărător (maximum 23 deplasări). Dacă după 23 deplasări nu a apărut un 1 în dreapta virgulei - mantisa este nulă şi se stabileşte rezultat nul.

g) Conţinutul numărătorului se introduce în poziţiile 2^4 ...2^0 ale exponentului.

h) Dacă s-au făcut deplasări la stînga acestea conduc la un exponent negativ şi se efectuează în acest caz complementul 2 al exponentului inclusiv poziţia pentru semn.

i) Se face corecţia exponentului pentru o polarizare corectă şi pentru stabilirea semnului rezultatului. Dacă caracteristica a fost pozitivă se adună 1 în poziţia 2^6 iar dacă caracteristica a fost negativă se adună 1 în poziţiile 2^7 şi 2^6.

j) Numărul obţinut se transferă într-un registru general al calculatorului.

In cazul operaţiei de logaritmare nu apar supradepăşiri sau subdepăşiri ale exponentului polarizat.

5.1.4. Algoritmul pentru antilogaritmare

Algoritmul pentru această operaţie este asemănător cu cel al operaţiei de logaritmare, doar succesiunea fazelor este inversată. Operaţia se poate aplica numerelor pozitive sau negative. Un număr pozitiv (o în poziţia 2^7) conduce la un exponent pozitiv (1 în poziţia 2^6) iar un număr negativ (1 în poziţia 2^7) conduce la un exponent negativ (o în poziţia 2^6) al rezultatului. Rezultatul operaţiei este întotdeauna un număr pozitiv. Etapele parcurse în ciclul de execuţie a instrucţiei sînt următoarele :

a) Se preia operandul dintr-un registru general sau din memoria calculatorului.

b) Dacă operandul este nul atunci antilogaritmul este 1 adică în virgulă flotantă olooooo1,1ooo..., rezultat ce se stabileşte direct. Se verifică semnul şi valoarea exponentului. Dacă exponentul este pozitiv şi cantitatea din poziţiile 2^5...2^0 este mai mare decît 6 se semnalizează supradepăşire în cazul operandului pozitiv sau subdepăşire în cazul operandului negativ. Cînd exponentul este negativ se efectuează complementul de 2 al exponentului polarizat inclusiv poziţia pentru semn. Dacă poziţiile 2^5... 2^0 conţin un număr mai mare decît 23 (numărul maxim de deplasări) se semnalizează subdepăşire şi se anulează rezultatul.

c) Poziţiile 2^4...2^0 ale exponentului polarizat - în cazul exponentului pozitiv - sau ale complementului de 2 al exponentului polarizat - în cazul exponentului negativ - se plasează într-un numărător, ele indicînd cantitatea deplasărilor ce trebuiesc efectuate. Sensul deplasărilor îl stabilește semnul pe care l-a avut exponentul operandului. In pozițiile $2^7 \ldots 2^0$ ale registrului operandului se înscrie 0.

d) Se efectuează deplasări ale părții fracționare a operandului după cum urmează :

- în cazul exponentului pozitiv se fac deplasări cu o poziție la stînga, reducînd de fiecare dată conținutul numărătorului cu 1, pînă la anularea lui (maximum 6 deplasări) ;

- în cazul cînd exponentul a fost negativ (înainte de efectuarea complementului de 2 al lui) se efectuează deplasări cu o poziție la dreapta, reducînd de fiecare dată conținutul numărătorului cu 1, pînă la anularea lui (maximum 23 deplasări).

e) In cazul cînd operandul a fost negativ se efectuează complementul de 2 al întregului operand inclusiv poziția pentru semn.

f_1) Biții din pozițiile $2^{-1} \ldots 2^{-24}$ ai operandului se trimit la intrările structurii logice celulare pentru generarea antilogaritmului. Aceasta se obține în mod asincron cu 23 de biți, care se deplasează în pozițiile $2^{-2} \ldots 2^{-24}$ iar în poziția 2^{-1} se înscrie un 1.

f_2) Se face corecția exponentului rezultatului pentru polarizare corectă și datorită mutării virgulei cu o poziție la stînga la generarea antilogaritmului. Dacă operandul a fost pozitiv și a avut exponent negativ se adună corecția ooooooo1. In celelalte cazuri se adună corecția oloooool.

g) Se verifică dacă în urma efectuării corecției a apărut o supradepășire (lo în pozițiile 2^7, 2^6) cînd se efectuează o derutare.

h) Se trimite rezultatul într-un registru general al calculatorului.

5.2. Structura dispozitivului aritmetic cu virgulă mobilă logaritmic.

Structura dispozitivului aritmetic cu virgulă mobilă a fost adoptată astfel încît ea să coincidă în cea mai mare parte cu structura unui dispozitiv aritmetic cu virgulă fixă astfel încît aceste două dispozitive să poată fi înlocuite printr-unul singur. Prin urmare dispozitivul nu va conține elemente separate pentru prelucrarea exponenților. Acest lucru este înlesnit și de faptul că operațiile cu logaritmi se efectuează cu caracteristica și

BUPT

mantisa neseparate.

Dispozitivul propus de autorul tezei (fig.5-3) constă din trei registre cu 32 poziţii binare considerate realizate cu bistabile master-slave, un numărător cu 5 poziţii binare şi structura logică celulară de generare a logaritmului şi antilogaritmului binar (cu anticipare, comasare şi asincronă).

Registrul acumulator A include şi circuitele pentru formarea sumei şi transportului simultan pe două nivele [55],[76]. Operandul I se poate aduce în registrul A dintr-un registru general al calculatorului [75],[76].

Registrul M poate primi operandul II din memorie sau dintr-un registru general. In operaţiile cu un singur operand acesta poate fi primit de registrul M numai din memorie.

Registrul B este necesar pentru păstrarea părţii fracţionare ce se logaritmează sau a mantisei ce se antilogaritmează şi pentru aplicarea lor fără întrerupere la intrările structurii logice celulare pînă la terminarea generării logaritmului sau antilogaritmului de către aceasta.

In fig.5-3 cu linie continuă s-a reprezentat circulaţia părţii fracţionare sau mantisei numerelor iar cu linie întreruptă circulaţia exponentului sau caracteristicii. O linie dublă continuă sau întreruptă reprezintă circulaţia perechilor de valori cu negaţiile lor (ambele ieşiri de la bistabile).

Structura logică celulară se consideră realizată cu circuite integrate în tehnologie rapidă cu t_p=6 ns, avînd 28 de coloane şi 28 de linii (Capitolul III).

Deoarece s-a urmărit determinarea duratei totale a operaţiilor în dispozitivul aritmetic cu virgulă mobilă, inclusiv ciclul de instrucţie, a fost necesar ca înainte de proiectarea acestuia să se stabilească în formă generală structura unităţii centrale a calculatorului şi structura cuvîntului instrucţie. Ele s-au adoptată asemănătoare cu cele ale calculatorului FELIX C-256 pentru ca în final să se poată face o comparaţie între dispozitivul aritmetic cu virgulă propus în teză şi operatorul în virgulă flotantă (OVF) al acestui calculator [75],[76].

Deasemenea orologiul şi registrul de faze al dispozitivului de comandă s-au considerat de tipul celor utilizate în calculatorul FELIX-256. Orologiul este realizat deci cu o linie de întîrziere ce poate funcţiona recurent în absenţa unei condiţionări, cu o perioadă T6=250 ns T9=350 ns şi condiţionat, în care caz perioada de recurenţă se adaugă la timpul de aşteptare a diferitelor

BUPT

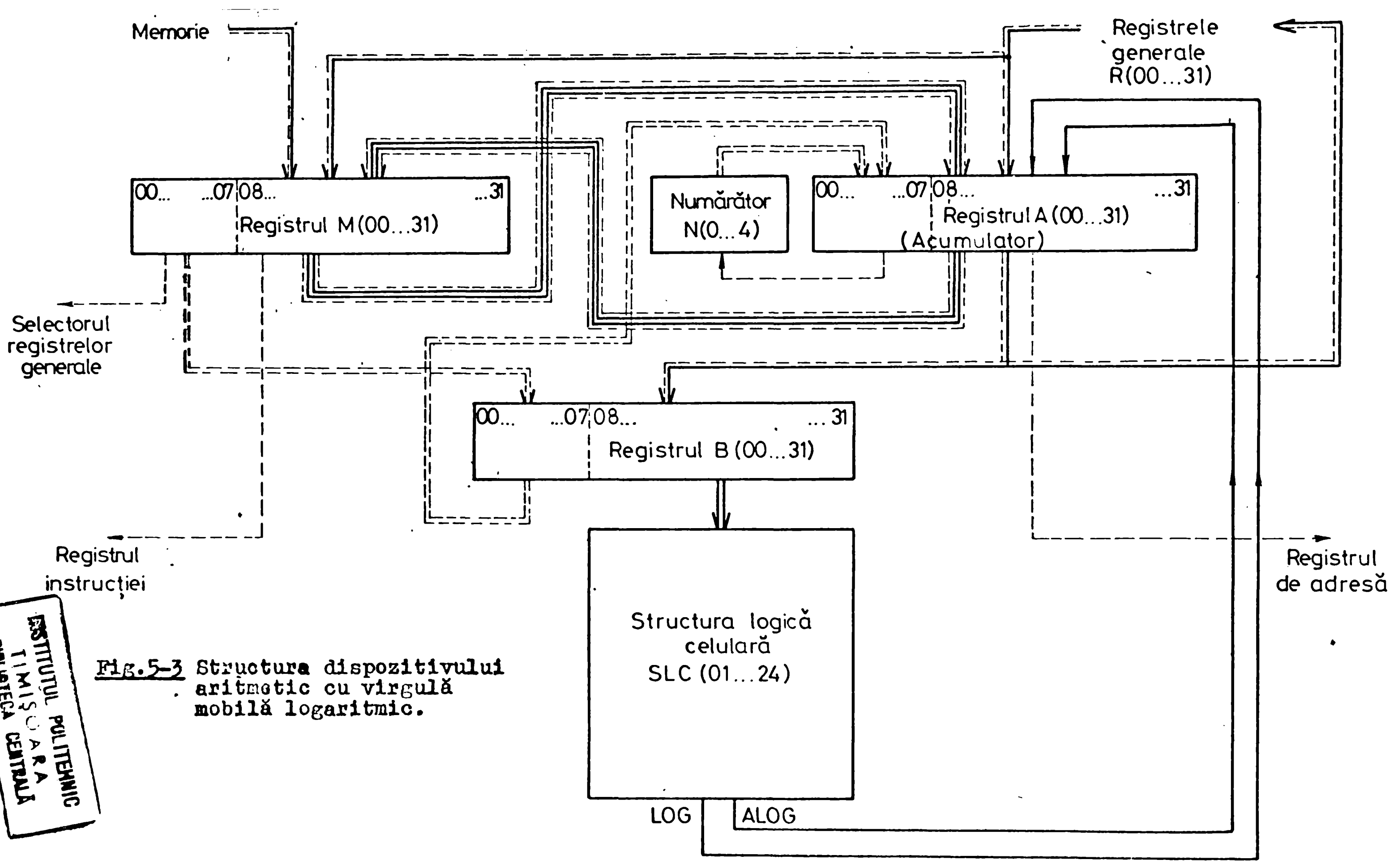


Fig.5-3 Structura dispozitivului aritmetic cu virgulă mobilă logaritmic.

INSTITUTUL POLITEHNIC
TIMIȘOARA
BIBLIOTECA CENTRALĂ

BUPT

condiţii provenind de la memorie sau de la structura logică celulară de logaritmare şi antilogaritmare. Impulsurile furnizate de orologiu au o durată de 4o ns.

Registrul de faze se consideră realizat dintr-un şir de bistabile master-slave comandate de impulsul de orologiu astfel încît la un moment dat un singur bistabil, corespunzător unei anumite "faze", este stabilit pe "1". Prin urmare numărul de bistabile este egal cu numărul de faze necesare la executarea unei instrucţii.

Faza reprezintă deci un interval de timp egal cu T6 sau T9 în care se poate efectua o microoperaţie. In prima parte a unei faze (21o ns sau 31o ns) pînă la apariţia impulsului de orologiu se stabilesc valorile logice ale intrărilor de comandă ale bistabilelor din registre sau din dispozitivul de comandă. La apariţia impulsului de orologiu informaţia se înscrie în bistabilele "master" iar la terminarea impulsului de orologiu informaţia se înscrie în bistabilele "slave".

Acest mod de lucru permite ca în aceeaşi fază să se transfere informaţia dintr-un registru în altul şi să se înscrie o nouă informaţie în primul registru.

Dispozitivul aritmetic cu virgulă mobilă propus a fost proiectat logic de către autorul tezei dar din cauza volumului mare al tabelelor şi ecuaţiilor proiectarea nu este inclusă în materialul de faţă.

La proiectarea dispozitivului aritmetic cu virgulă mobilă logaritmic s-a urmărit realizarea unor durate cît mai mici ale celor 8 operaţii, adică realizarea acestora într-un număr cît mai redus de faze. Acesta a fost criteriul de bază în proiectare. Din acest motiv numai primele 7 faze ale tuturor celor 8 instrucţii au putut fi făcute identice şi cuprinse deci într-o secvenţă comună. Aceasta reprezintă ciclul instrucţiei.

Ciclul de execuţie al instrucţiei conţine fazele 08...21 pentru adunare, scădere, înmulţire şi împărţire, o8...19 pentru ridicare la patrat, extragerea rădăcinii patrate, antilogaritmare şi o8...18 pentru logaritmare. Intrucît ultima fază a fiecărei instrucţii este identică este posibil să se considere încă o secvenţă comună incluzînd această fază.

La stabilirea microoperaţiilor executate în fiecare fază a instrucţiilor ce utilizează logaritmii s-a ţinut cont de modul special în care se lucrează cu structura logică celulară de generare a logaritmului şi antilogaritmului asincronă (Capitolul III).

Mai întîi la intrările structurii trebuie aplicat numărul

BUPT

zero cu un timp $t > t_{omax} = 312$ ns înainte de aplicarea operandului, pentru stabilirea stării iniţiale celei mai avantajoase. Prin urmare comanda de stabilire la "0" a intrărilor structurii se va da cu o fază lungă (T9) sau două fase scurte (T6) mai înainte de aplicarea operandului la intrări.

Testarea semnalului $ID_{29,24}$ de terminare a descompunerii în structura logică celulară se face numai după un timp $t > t_{omax} = 312$ ns, adică o fază lungă (T9) de la aplicarea operandului la intrările structurii. Orologiul dispozitivului de comandă se va opri pînă la apariţia semnalului $ID_{29,24}$ cînd se reporneşte. La repornire faza în curs se continuă cu încă o perioadă T6, timp suficient pentru stabilirea la valorile corecte a ieşirilor pentru logaritm sau antilogaritm ale structurii logice celulare deoarece:

$$T6 - t_H = 210 \text{ ns} \quad (n+2)t_p = 180 \text{ ns} \qquad (5-6)$$

unde $t_H = 40$ ns este durata impulsului de orologiu. Prin urmare deja la sfîrşitul aceleiaşi faze se poate prelua şi utiliza logaritmul sau antilogaritmul de la structura logică celulară.

Durata microoperaţiei de adunare la un sumator pentru numere cu 32 de biţi cu transport simultan cu două nivele, utilizînd circuite integrate cu $t_p = 6$ ns, rezultă de o valoare sub T6 (aproximativ 100 ns) astfel că ea se poate realiza într-o singură fază.

Duratele celor opt operaţii realizate de dispozitivul aritmetic cu virgulă mobilă proiectat au rezultat cuprinse între limitele date în Tabelul 5-1. Tot acolo se dau şi duratele medii ale operaţiilor determinate prin considerarea mediei aritmetice a timpilor maximi şi minimi de generare a logaritmului şi antilogaritmului (aici de terminare a descompunerii) şi a timpilor minim şi maxim de normalizare. S-a ţinut cont de suprapunerea unor faze în timp cu efectuarea descompunerii în structura logică celulară. În realitate durata medie a operaţiilor de descompunere în structura logică celulară este mai mică decît cea obţinută prin medie aritmetică deoarece cazurile de durată apropiată de cea maximă sînt extrem de puţine.

Duratele din tabelul 5-1 au fost stabilite în următoarele condiţii :

- adresarea la operandul II sau la operandul unic este directă iar operandul este în memorie,
- cererea de acces la memorie este satisfăcută în timpul de acces de 350 ns [75],
- operanzii sînt diferiţi de zero,
- nu apar subdepăşiri ale exponentului rezultatului,

BUPT

- nu se produc derutări,
- durata fazelor scurte este de 250 ns iar a celor lungi 350 ns,
- se include şi ciclul instrucţiei care durează 2,1 μs.

Tabelul 5-1

Instrucţia	Cod	Durata minimă	Durata maximă	Durata medie
Adunare sau Scădere	AD SC	5,95 μs	11,45 μs	8,70 μs
Inmulţire sau Impărţire	INM IMP	6,00 "	14,15 "	9,60 "
Ridicare la patrat sau Extragerea rădăcinii patrate	RID EXTR	5,65 "	10,95 "	7,95 "
Logaritmare	LOG	5,30 "	13,65 "	9,30 "
Antilogaritmare	ALOG	5,55 "	13,50 "	9,35 "

Faţă de un dispozitiv aritmetic cu virgulă mobilă obişnuit dispozitivul logaritmic are o serie de particularităţi şi avantaje ca :

- folosirea unei structuri asemănătoare cu cea a unui dispozitiv aritmetic cu virgulă fixă fiind posibilă suprapunerea celor două dispozitive,
- folosirea unui număr mare de comenzi ce apar la dispozitivul cu virgulă fixă,
- asigurarea unor durate medii a operaţiilor aproximativ egale şi mai mici,
- cantitatea fazelor necesare la toate cele 8 instrucţii este aproximativ aceeaşi şi nici o instrucţie nu necesită reluarea întregii succesiuni de faze,
- lipsa necesităţii normalizării şi rotunjirii la înmulţire şi împărţire,
- suprapunerea microoperaţiilor de citire din memorie a operandului II şi de generare a logaritmului operandului I ceea ce reduce durata înmulţirii şi împărţirii.

In fig.5-4 se prezintă registrul acumulator A sub formă de bloc cu borne de intrare şi ieşire [61]. La acesta vin comenzi de la dispozitivul de comandă şi ieşiri de la registrele M,R,B,N şi de la structura logică celulară. Spre alte registre şi spre dispozitivul de comandă pleacă perechile de ieşiri ale bistabilelor A00 ... A31.

BUPT

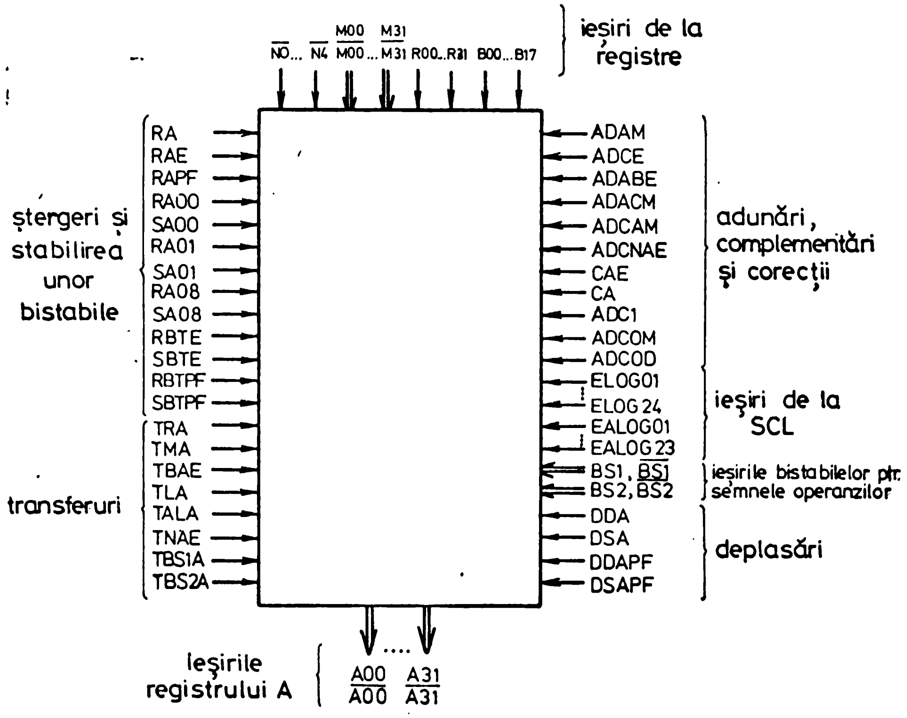


Fig.5-4. Blocul registrului acumulator A.

Numărul relativ mare de comenzi ale registrului A se datorează în special faptului că se prelucrează în acelaşi registru şi exponentul şi partea fracţionară a numerelor în forma cu virgulă mobilă. Blocul acumulatorului mai include două bistabile pentru transport în poziţiile 2^0 şi 2^{-24} cu ajutorul cărora se poate simplu aduna un bit 1 în aceste poziţii la efectuarea complementului de 2.

In fig.5-5, 5-6 şi 5-7 se prezintă sub formă de blocuri cu borne de intrare şi ieşire registrele M, B şi numărătorul N.

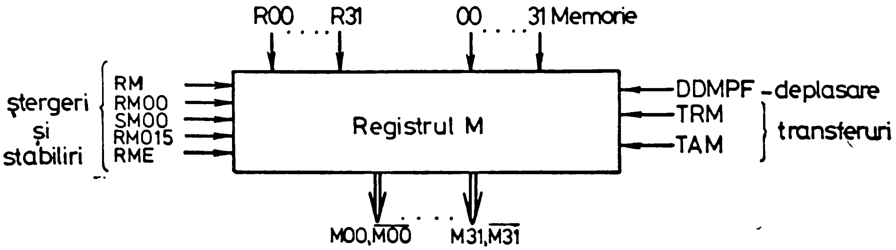


Fig.5-5. Blocul registrului M.

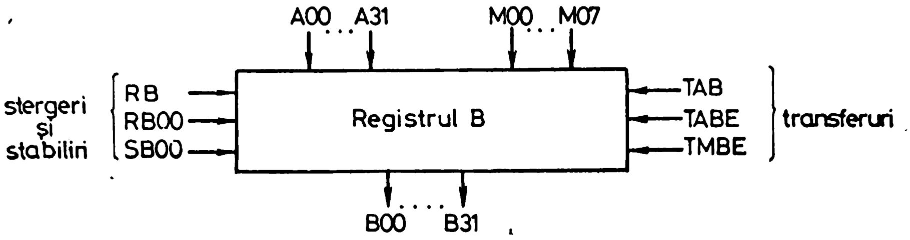


Fig.5-6. Blocul registrului B.

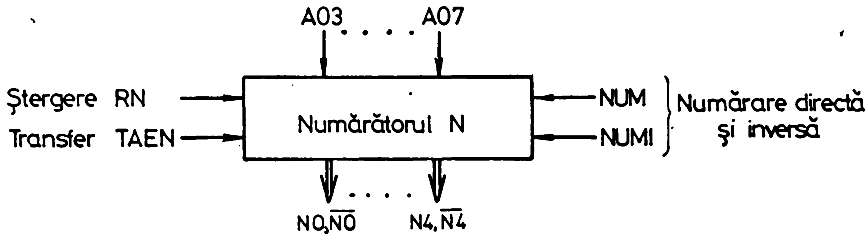


Fig.5-7. Blocul numărătorului N.

In fig.5-8 se prezintă sub formă de bloc cu borne de intrare şi ieşire structura logică celulară pentru generarea logaritmului şi antilogaritmului.

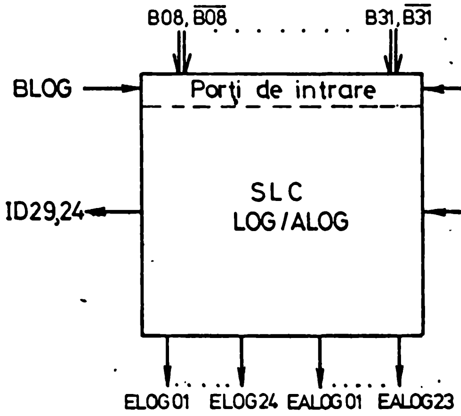


Fig.5-8. Blocul SLC

Pentru generarea logaritmului se dau comenzile BLOG=1 şi BILOG=1 iar pentru generarea antilogaritmului BALOG=1.

La generarea antilogaritmului se utilizează numai 23 de biţi la ieşirea SLC care se plasează în dreapta unui 1 din poziţia 2^{-1} a acumulatorului deci precizia de calcul este mai bună

BUPT

decît s-a estimat anterior.

Folosirea unui dispozitiv de comandă local a dispozitivului aritmetic cu virgulă mobilă logaritmic nu este avantajoasă din cauza numărului mare de conexiuni dintre acesta şi dispozitivul de comandă central. Problema dispozitivului de comandă local intervine inevitabil acolo unde dispozitivul aritmetic cu virgulă flotantă este o opţiune pentru un calculator numeric universal.

Comparînd dispozitivul aritmetic cu virgulă mobilă logaritmic propus în acest capitol cu un dispozitiv de tip obişnuit, ca de exemplu operatorului virgulă flotantă al calculatorului FELIX C-256 [75],[76] care este un dispozitiv de performanţă, rezultă următoarele concluzii generale privind cantitatea circuitelor şi viteza operaţiilor :

- dispozitivul logaritmic propus conţine în plus structura logică celulară de generare a logaritmului şi antilogaritmului, restul circuitelor, inclusiv cele pentru comanda dispozitivului aflate în dispozitivele de comandă, fiind comparabile ca extindere.
- precizia de calcul cu lungime simplă de cuvînt este practic aceeaşi ;
- durata operaţiilor de adunare şi scădere este aproximativ aceeaşi la ambele dispozitive [74] ;
- dispozitivul logaritmic realizează înmulţirea într-un timp mediu de 9,6 μs faţă de 23,7 μs de la dispozitivul obişnuit [74] adică de cca 2,5 ori mai repede şi împărţirea într-un timp mediu de 9,6 μs faţă de 28,8 μs de la dispozitivul obişnuit adică de 3 ori mai repede.
- dispozitivul logaritmic poate realiza în plus patru operaţii : ridicare la patrat, extragerea rădăcinii patrate, logaritmarea şi antilogaritmarea iar cu ajutorul acestora se fac simplu încă o serie de operaţii ca ridicarea la o putere de orice ordin, extragerea unei rădăcini de orice ordin, calculul logaritmilor şi antilogaritmilor în orice bază, adică operaţii care se efectuează prin subrutine în dispozitivul obişnuit, într-un timp relativ lung.

Singurul dezavantaj al dispozitivului logaritmic este deci acela al necesităţii unei structuri logice celulare de dimensiuni mari, cu un număr mare de circuite logice. Intr-adevăr, la realizarea structurii cu circuite integrate TTL standard cantitatea de capsule necesare este foarte mare. In cazul utilizării unei tehnologii LSI şi ELSI dimensiunile structurii şi cantitatea capsulelor pot deveni însă acceptabile.

BUPT

CONCLUZII

In cadrul primei părţi a tezei (Capitolele I, II, III, IV) s-a arătat posibilitatea generării logaritmilor şi antilogaritmilor binari prin hardware obţinîndu-se o soluţie bazată pe descompunerea în factori şi termeni, cu următoarele performanţe :

- schema electronică constă dintr-o structură logică celulară care prezintă avantajele tipice acestor structuri şi care operează cu numere normalizate în domeniul [1, 2),
- durata generării logaritmului şi antilogaritmului binar, pentru o cantitate de biţi n= 28, este cuprinsă între o...532 t_p respectiv între o...463 t_p (unde t_p este timpul de propagare tipic pe un nivel al circuitelor logice utilizate) situaţie ce se exploatează printr-o funcţionare asincronă a schemei,
- eroarea logaritmului sau antilogaritmului se poate face oricît de mică prin simpla creştere a numărului de celule deoarece ea afectează numai ultimii 4-5 biţi.

Soluţia propusă de autorul tezei elimină sau atenuează desavantajele soluţiilor propuse de alţi autori, comentate în Capitolul I al tezei.

In măsura în care integrarea pe scară largă şi extralargă va permite realizarea în viitor, la dimensiuni acceptabile, a structurii logice celulare de generare a logaritmilor şi antilogaritmilor binari cu anticipare şi asincronă, soluţia propusă de autorul tezei va putea fi aplicată în instalaţii de calcul specializate, în calculatoare electronice de birou şi în calculatoarele electronice universale pentru calcule cu numere fracţionare, permiţînd efectuarea prin cablaj a unui număr mare de operaţii în plus faţă de cele 4 operaţii clasice. Soluţia propusă în teză constituie deci un pas înainte în desvoltarea firmware-ului calculatoarelor numerice şi deci încă un pas spre generaţia a IV-a de calculatoare.

Pentru demonstrarea posibilităţilor unei structuri logice celulare de generare a logaritmilor şi antilogaritmilor binari s-a conceput în Capitolul V un dispozitiv aritmetic cu virgulă mobilă logaritmic, care operează pe basa unor algoritmi speciali şi care poate efectua rapid 4 operaţii aritmetice în plus faţă de cele obişnuite şi anume :

- ridicarea la patrat,
- extragerea rădăcinii patrate,
- logaritmarea în basa 2,
- antilogaritmarea în basa 2,

BUPT

ultimele două constituind o posibilitate de a executa în continuare, cablat sau prin program, operații de ridicare la orice putere, extragere a rădăcinii de orice ordin, calculul logaritmilor și antilogaritmilor în orice bază.

Conținînd în plus față de un dispozitiv aritmetic cu virgulă mobilă clasic doar structura logică celulară, dispozitivul aritmetic logaritmic poate efectua înmulțirea și împărțirea de 2,5 respectiv 3 ori mai repede decît cel clasic. Inafară de aceasta, dispozitivul aritmetic logaritmic cu virgulă mobilă conceput mai are o serie de avantaje ca :

- posibilitatea utilizării acelorași registre ca și dispozitivul aritmetic cu virgulă fixă,
- folosirea unui număr mare de comenzi de la dispozitivul aritmetic cu virgulă fixă.

Contribuțiile autorului

In cadrul tezei autorul a adus o serie de contribuții în domeniul aritmeticii și al structurii dispozitivelor aritmetice ale calculatoarelor numerice.

Principalele contribuții, în ordinea în care apar în teză, sînt următoarele :

1. Analiza critică și comparativă a algoritmilor existenți de calcul al logaritmilor și antilogaritmilor, stabilirea duratei operațiilor în cazul celui de al treilea algoritm Dean, semnalarea incorectitudinii algoritmului Perle și indicarea unei soluții de corectare a acestuia (Capitolul I.).

2. Elaborarea unui algoritm nou de calcul al logaritmilor și antilogaritmilor binari bazat pe înmulțiri și împărțiri cu constante de forma $C_i = 2^{2^{-i}}$, utilizabil la generarea prin hardware a logaritmilor și antilogaritmilor binari (Capitolul II).

3. Elaborarea unui algoritm nou de calcul al logaritmilor și antilogaritmilor binari bazat pe descompunerea în factori de forma $(1+2^{-i})$ și termeni de forma $L_i = \log_2(1+2^{-2})$, care necesită numai operații de adunare, deplasare și comparare. Analiza erorilor algoritmului și a posibilităților de reducere a acestora (Capitolul II).

4. Calculul constantelor L_i și C_i care intervin în cei doi algoritmi, cu ajutorul calculatorului electronic cu 43 cifre binare exacte (13 cifre zecimale), utile la operații cu numere avînd o cantitate mare de biți (Capitolul II).

BUPT

5. Concepţia unui dispozitiv de generare a logaritmilor şi antilogaritmilor binari care utilizează algoritmul bazat pe înmulţiri şi împărţiri cu constante şi care este mai simplu şi mai rapid decît un alt dispozitiv similar cunoscut în literatură. Dispozitivul este aplicabil în calculatoarele în care operaţiile de înmulţire şi împărţire se efectuează în mod obişnuit sau printr-o structură logică celulară cînd este necesar în plus doar un generator de constante C_i (Capitolul III).

6. Elaborarea unei metode de proiectare a generatoarelor de constante C_i şi L_i constînd dintr-un registru de deplasare cu modificarea stării, care necesită o cantitate de circuite logice minimă şi care este uşor de realizat cu circuite integrate TTL. Generatoarele sînt necesare atunci cînd algoritmii elaboraţi de autor se implementează pe un dispozitiv aritmetic obişnuit (Capitolul III).

7. Concepţia unei structuri logice celulare rapide de generare a antilogaritmilor binari bazată pe înmulţirea constantelor $C_i = 2^{2^{-i}}$, utilizabilă în unele instalaţii de calcul specializate (Capitolul III).

8. Concepţia unei structuri logice celulare de generare a logaritmilor şi antilogaritmilor binari bazată pe descompunerea în factori de forma $(1+2^{-i})$ şi termeni $L_i = \log_2(1+2^{-i})$, singura cunoscută în literatura de specialitate.

Structura a fost concepută în mai multe variante :

- structuri logice simple separate,
- structură logică simplă cu comasare,
- structuri logice cu anticipare separate,
- structură logică cu anticipare şi comasare,
- structură logică cu anticipare şi comasare asincronă.

Ultima din acestea, generează logaritmul sau antilogaritmul într-un timp cuprins între 0 şi o valoare maximă sub $m^2 t_p$ (unde m este cantitatea de biţi a operanzilor în calculator iar t_p este timpul de propagare tipic pe un nivel logic din celulele structurii). Deci generarea durează un timp mediu mai mic decît $\frac{1}{2} m^2 t_p$, timpul cel mai scurt realizat pînă în prezent de un dispozitiv de generare a logaritmilor şi antilogaritmilor. Structura este utilizabilă într-un dispozitiv aritmetic cu virgulă mobilă (Capitolul III).

9. Elaborarea unui program de simulare pe calculator numeric a unei scheme logice combinaţionale bazat pe urmărirea selectivă cu aplicaţie specială la simularea structurilor logice celulare (Capitolul IV).

BUPT

10. Elaborarea algoritmilor pentru 6 operaţii aritmetice care utilizează logaritmii, dintr-un dispozitiv aritmetic cu virgulă mobilă ce dispune de o structură logică celulară de generare a logaritmilor şi antilogaritmilor binari (Capitolul V).

11. Concepţia şi proiectarea unui dispozitiv aritmetic cu virgulă mobilă logaritmic care poate efectua 8 operaţii aritmetice (adică 4 operaţii în plus faţă de un dispozitiv obişnuit) şi care are următoarele performanţe :

- realizează înmulţirea şi împărţirea într-un timp de 35 respectiv de 3 ori mai mic decît un dispozitiv obişnuit,
- poate folosi aceleaşi registre şi o parte din comenzi ca şi un dispozitiv aritmetic cu virgulă fixă (Capitolul V).

BUPT

BIBLIOGRAFIE

1. Mitchell J.N., Computer multiplication and division using binary logarithms. IRE Transactions on Electronic Computers, EC 11, No.4,1962, p.512-517.

2. Meggit J.E. Pseudo division and pseudo multiplication processes. IBM Jurnal, April 1962, p.210-226.

3. Ginkin G.G., Logarifmî, deţibelî, deţiloghi. Gosudarstvennoe Energeticeskoe Izdatelistvo, Moskva 1962, p.256-260.

4. Larsen R.P., Mavo M.M., Modeling and simulation of digital network. Communications of ACM, Vol.8, May 1965, p.308-312.

5. Combet H., Van Zonneveld, Verbeek L., Computation of the base two logarithm of binary numbers. IEEE Transection on Electronic Computers, EC 14, No.6, December 1965, p.863-867.

6. Dean K.J., Binary logarithms. Electronic Engineering, Vol.40, No.488, 1968, p.560-562.

7. Dean K.J., Design of bimary logarithms generators. Proceedings of the IEE, No.115, 1968, p.1118-1120.

8. Hoffmann,J.C.,Lacase B.,Csillag P., Iterative logical network for parallel multiplication. Electronics Letters, Vol.4, No.9, 1968, p.178.

9. Kants W.H., Levitt K.N.,Wakeman A.,Cellular interconection arrays. IEEE Transactions on Computers, Vol.C 17, No.5, May 1968.

10. Burton D.P., Noaks D.R., High - speed iterative multiplier Electronics Letters, Vol.4, No.13, 1968, p.262.

11. Dean K.J., Bimary division using a data dependent iterative arrays. Electronics Letters, Vol.4, No.14, Iune 1968, p.283-284.

12. Dean K.J., Cellular logical array for extracting square roots. Electronics Letters, Vol.4, No.15, 1968, p.314-315.

BUPT

13. Dean K.J., Versatile multiplier arrays. Electronics Letters, Vol.4, No.16, 1968, p.333-334.

14. Dean K.J., Design for a full multiplier. Proceedings of the IEE, Vol.115, No.11, November 1968, p.1592-1594.

15. Dean K.J., Iterative arrays of logical circuits for performing arithmetic. Electronic Engineering, Vol.4o, No.49o, December 1968, p.694-697.

16. Rammamoorthy C.V., Economides S.C., Fast multiplication cellular arrays for LSI implementation. 1969 Fall Joint Computer Conf., AFIPS Proceedings, Vol. 35, p.89-98.

17. Andrews C.A., Algorithm for finding logarithms of binary numbers to the base two. IBM Technical Disclosure Bulletin, Vol.11, No.8, January 1969, p.914-916.

18. De Mori R., Sugestion for an IC fast parallel multiplier, Electronics Letters, Vol.5, No.3, February 1969, p.5o-51.

19. Dean K.J., A fresh approach to logarithmic computation. Electronic Engineering, Vol.41, No.494, April 1969.

2o. Philo P.W., An algorithm to evaluate the logarithm of a number to base 2 in binary form. The Radio and Electronic Engineer, Vol.38, 1969, p.49-50.

21. Dean K.J., Some applications of cellular logic arithmetic arrays. The Radio and Electronic Engineer, Vol.37, No.4, April 1969, p.225-227.

22. Guild, H.H., Fully iterative fast array for binary multiplication and adition. Electronic Letters, Vol.5, No.12, Iune 1969, p.263.

23. Dean K.J., Cellular logical array for obtaining the square of a binary number. Electronics Letters Vol.5, No.16, August 1969, p.37o-371.

24. Perle M.D., The dual logarithm algorithm. Computer Design, Vol.9, No.1, Ianuary 197o, p.88-9o.

25. Hall, E.L., Lynch D.D., Dwyer S.J., Generation of products and quotients using approximate binary logarithms for digital filterring applications. IEEE Transactions on Electronic Computers Vol.C 19, No.2, February 1970, p.97-105.

26. Logan J.R., A design technique for digital squaring network. Computer Design, Vol.9, No.2, February 1970, p.84-88.

27. Guild H.H., Some cellular logical arrays for nonrestoring binary division. The Radio and Electronic Engineer, Vol.39, No.6, Iune 1970, p.345-348.

28. Florine J., Calculateurs numériques célulaires. Automatisme, Vol.15, No.718, Julliet-August 1970, p.327-330.

29. Guild H.H., Cellular logical array for nonrestoring square root extractions. Electronics Letters, Vol.6, No.3, 1970, p.66-67.

30. De Vries R.C., Chao M.H., Fully iterative array for extracting square roots. Electronics Letters, Vol.6, No.8, 1970, p.255-256.

31. Fréoon L., Multiplicateur cellulaire parallele des nombres en virgule flottante. Electronics Letters, Vol. 6, No.8, 1970, p.226-228.

32. White G., Generalised cell for use in iterative and near iterative arithmetic arrays. Electronics Letters, Vol.6, No.9, 1970, p.270-271.

33. Majithia J.C., Nonrestoring binary division using a cellular array. Electronics Letters, Vol.6, No.10, 1970, p.303-304.

34. Walker P.A,W., Asincron binary division array. Electronics Letters, Vol.6, No.16, 1970, p.515-517.

35. Soceneanțu A., Dispozitiv cablat sub formă de matrice pentru realizarea împărțirii binare. Dosar 65911/1970, OSIM București.

36. Cingudean M., Matrice logică iterativă pentru ridicarea la patrat a numerelor binare. Sesiunea de comunicări științifice în domeniul calculatoarelor electronicii și automaticii, IPT. Mai 1970.

BUPT

37. Ciugudean M., Structuri celulare de logaritmare şi antilogaritmare pentru un dispozitiv aritmetic cu virgulă mobilă. Referat nr.3 pentru doctorat, septembrie 1970.

38. Pop V., Schemă logică iterativă pentru efectuarea înmulţirii binare. Buletinul Ştiinţific şi Tehnic al I.P.T. Seria electrotehnică, Tom 15, fasc.2, 1970, p.233-238.

39. Habibi A., Wintz P.A., Fast multipliers. IEEE Transactions on Computers, Vol.C 19, No.2, February 1970, p.153-157.

40. Krausener I.M., Description et aplication d'une unité arithmétique a MSI - utilisant les circuits SN 74-181/182. Inter Electronique, Vol.25, Nr.12, Decembre 1970, p.38-45.

41. Pezaris S.D., A 40 ns-17 bit by 17 bit array multiplier IEEE Transactions on Computers, Vol.C, February 1971, p.462-447.

42. Dean K.J., Cellular multiplier subarrays : a critical path approach to propagation times. Electronics Letters, Vol.7, No.3, 1971, p.75-77.

43. Gardiner A.B., Hont J., Comparaisson of restoring and non restoring cellular array dividers. Electronics Letters, Vol.7, No.8, 1971, p.172-173.

44. Jullien, G.A., Fast algorithm for digital logaritmic conversion. Electronics Letters, Vol.7, No.9, 1971, p.218-220.

45. Nicoud J.D., Dessoulavy R., Logaritmic convertor. Electronics Letters, Vol.7, No.9, 1971, p.230-231.

46. Guild H.H., Fast versatile binary comparator array. Electronics Letters, Vol.7, No.9, 1971, p.225-226.

47. Devereel J., Multiplication of complex numbers using iterative arrays. Electronics Letters, Vol.7, No.9, 1971, p.205-207.

48. Kingsbury N.G., High-speed binary multiplier. Electronics Letters, Vol.7, No.10, 1971, p.277-278.

49. Frécon L., Réseau logique cellulaire pour l'extraction

BUPT

des racines carrées (virgule flottante). Electronics Letters, Vol.7, No.13, 1971, p.361-362.

50. Majithia J.C., Siemens K.H., Comment on the speed of binary multiplication using cellular arrays. Electronics Letters, Vol.7, No.15, 1971, p.430-431.

51. Gex A., Multiplier-divider cellular array. Electronics Letters, Vol.7, No.15, 1971, p.442-444.

52. Gardiner A.B., Asynchronous binary restoring divider array. Electronics Letters, Vol.7, No.18, 1971, p.542-544.

53. Deverel J., Generation of sines and cosines using iterative arrays. Electronics Letters, Vol.7, No.20, 1971, p.616-618.

54. Deegan I., Concise cellular array for multiplication and division. Electronics Letters, Vol.7, No.23, 1971, p.702-704.

55. Dancea I. Dispozitivele aritmetice ale calculatoarelor numerice. Editura Tehnică, Bucureşti 1971.

56. Ciugudean M., Circuit logic iterativ de antilogaritmare. Prima sesiune de comunicări ştiinţifice a tinerilor ingineri şi cercetători studenţi din Timişoara. Secţia Calculatoare-electronică. Dec. 1971, p.57-66.

57. Boyce A.H., Emmerson B.G., Stringer D.V., West B.G., Simmulation of binary logic circuits by digital Computers. Elspress Informatia. Vîcislitelinaia Tehnika, No.37, 1971.

58. Ciugudean M., Algoritm de calcul al logaritmilor şi antilogaritmilor binari utilizabil în calculatoarele electronice. Automatica şi Electronica, Nr.3, 1972, p.105-109.

59. Soceneanţu A., Toma C.I., Cellular logic array for redundant binary division. Proceedings of the IEE, Vol. 119, No.10, October 1972.

BUPT

60. Ralph T., High speed binary multiplication using the MC 10181. Application note AN-566. Motorola Semiconductor Products Inc.,1972.

61. Rogojan A., Metoda pentru sinteza schemei logice a unui calculator numeric. Teză de doctorat. Institutul Politehnic Bucureşti, 1974.

62. Marino D., New algorithms for the approximate evaluation in hardware og binary logarithms and elementary functions. IEEE Transactions on Computers, December, 1972, Vol.C 21, No.12, p.1416-1421.

63. Pop V., Schemă logică iterativă pentru înmulţirea şi împărţirea binară. Automatica şi Electronica No.1, 1973, p.37-43.

64. Ciugudean M., Program de simulare a unei scheme logice combinaţionale bazat pe urmărirea selectivă. Buletinul Stiinţific şi Tehnic al IPT. Seria electrotehnică. Tom 18, Fasc.1/1973, p.49-58.

65. Schmid H., BCD Logic. Electronic Design, June 1973.

66. Ciugudean M., Algoritm şi schemă logică pentru calculul logaritmilor şi antilogaritmilor binari bazate pe înmulţiri şi împărţiri cu constante. Automatica şi electronica, Nr.6, 1973, p.278-282.

67. Toma C.I., Cellular logic array for multiplication of signed binary numbers. Report No.6, North Carolina State University. Department of electrical engineering, November 1973.

68. Compagnie International pour l'Informatique, Manuel d'utilisation IRIS 50.

69. Texas Instruments Inc. Integrated Circuits 1971.

70. Ciugudean M., Popescu V., Generator de constante pentru calculul logaritmilor şi antilogaritmilor binari în calculatoarele electronice. Buletinul Ştiinţific al I.P.T. Fasc.1, 1975.

71. Ciugudean M., Algoritm pentru un operator aritmetic cu virgulă flotantă logaritmic. Sesiunea de comunicări ştiinţifice în domeniul calculatoarelor IPT, mai 1974 (în curs de publicare).

BUPT

72. Ciugudean M., Algorithm for computing logarithms and antilogarithms. Computer Design, July 1974, p.106.

73. Ciugudean M., Determinarea unor şiruri de constante utilizate la generarea logaritmilor şi antilogaritmilor binari în calculatoarele electronice. Studii şi cercetări matematice. Academia RSR (în curs de publicare).

74. I.T.C.Bucureşti, Programul NANO pentru calculatorul FELIX C-256.

75. Epure M., şi alţii, Calculatoarele FELIX C-256, IRIS-50, IBM 360/30,40. Funcţionare şi depanare. Ed.Tehnică, 1974.

76. C.I.I. Manual de utilizare al calculatorului IRIS-50.

BUPT