(19) LIETUVOS RESPUBLIKA VALSTYBINIS PATENTŲ BIURAS V P B

(10) **LT IP382 A**

(12) **PARAIŠKOS APRAŠYMAS**

(21) Paraiškos numeris: **IP382**

(22) Paraiškos padavimo data: **1993 03 05**

(41) Paraiškos paskelbimo data: **1994 11 25**

(62) Paraiškos, iš kurios dokumentas išskirtas, numeris: —

(86) Tarptautinės paraiškos numeris: —

(86) Tarptautinės paraiškos padavimo data: —

(85) Nacionalinio PCT lygio procedūros pradžios data: —

(30) Prioritetas: **9002558, 1990 08 02, SE**
PCT/SE91/00516, 1991 08 01, SE

(51) Int. Cl. (2006): **G06F 7/00**
G06F 7/38
G06F 7/48
G06F 9/302
G06F 9/44
G06F 12/00
G11C 11/412
G11C 11/417
G11C 11/419
G11C 15/00

(71) Pareiškėjas:
CARLSTEDT ELEKTRONIK AB, Industrivagen 55, S-433 61 Partille, SE

(72) Išradėjas:
Lars Gunnar CARLSTEDT, SE

(74) Patentinis patikėtinis/atstovas:
Reda ŽABOLIENĖ, Advokatės Redos Žabolienės kontora METIDA, Verslo centras VERTAS, Gynėjų g. 16, LT-01109 Vilnius, LT

(54) Pavadinimas:

Redukuojantis procesorius

(57) Referatas:

—

LT IP382 A

Redukuojantis procesorius

Šis išradimas liečia redukuojantį procesorių.

IŠRADIMO PAGRINDIMAS

Kompjuteriai buvo išrasti 1940 metais. Nuo to laiko jie vystėsi revoliuciniu greičiu. Nežiūrint į tai, šių dienų kompjuteriai turi beveik tokią pačią architektūrą, kaip ir pirmieji.

Daug pasiekta skaitmeninės technikos srityje. VLSI pasirodymas ir litografijos pažanga leidžia pagaminti kompjuterius, sudarytus iš vienos skaitmeninės mikroschemos, kurios galimybės prilygsta penkių metų senumo superkompjuterio galimybėms. Matmenys sumažėjo eksponentiškai ir dabar mažesni negu 10^{-6} m. Taktinis dažnis ir tranzistorių skaičius išaugo eilę kartų. Fiziniai apribojimai riboja linijos plotį ne mažesnį kaip $2*10^{-6}$ m.

Tuo pat metu kompjuterių architektūra nepatobulėjo silicinių elementų panaudojime. Atvirkščiai, didesniam veikimo greičiui pasiekti daugelyje kompjuterių naudojama žymiai daugiau silicio elementų, negu jų reikia optimaliam darbui.

Šių dviejų faktų egzistavimas ateinančius penkis metus stabdys tradicinių procesorių darbo greičio didinimo galimybes. Pasirodžiusius lygiagrečius procesorius charakterizuoja padidėjusi dėl savo sudėtingumo elektroninės įrangos kaina bei, kas liečia daugelį programų tipų, pernelyg aukštos programinės įrangos kainos.

Palyginus tarpusavyje, elektroninės įrangos kaina nukrito, tuo tarpu kai naujų sistemų programinio aprūpinimo kaina žymiai išaugo ir pasiekė pernelyg aukštą lygį.

Kompjuteris susideda iš programinių ir elektroninių komponentų visumos. Skirtingos paradigmos ir vystymosi lygiai įvedė standartus - specializuotus ir plačiai paplitusius - kurie išsivystė į sistemą. Dėl šio nestandartiškumo ir atsirado didelis skirtingų interfeisų tipų skaičius.

Visi šie skirtingos kokybės ir tipo interfeisai bei paradigmos padarė kompjuterį sunkiai prieinamą tiek naudotojui, tiek ir programuotojui, kadangi jo naudojimas reikalauja pakankamai daug žinių, ko pasekoje programuotojas dėl sistemos sudėtingumo gali įvelti daug paslėptų klaidų.

Pastaruoju metu buvo pasirodė taip vadinami redukuojantys procesoriai (procesoriai su sumažintu instrukcijų skaičiumi). Redukuojantis procesorius vykdo tam tikros struktūros, įjungiančios aritmetines išraiškas, programą, ir vykdymo metu ta struktūra yra minimizuojama (redukuojama) atitinkamais tam tikrais būdais. Tokiu būdu programa nėra vykdoma nuosekliu būdu, kaip kitų tipų kompjuteriuose.

Tačiau iškyla tam tikri sunkumai konstruojant redukuojantį procesorių su didesnėmis galimybėmis.

Programavimo kalbų vystymasis

Pirmųjų elektroninių kompjuterių vystymasis iniciavo pritaikytų šiems kompjuteriams kelių programavimo kalbų, kaip FORTRAN, COBOL, Algol, BASIC, Pascal, atsiradimą ir vystymąsį. Šios kalbos buvo pavadintos bendru imperatyvinių kalbų terminu, arba toliau minimu tradicinių kalbų terminu, dėl to, kad šiomis kalbomis parašytos programos susideda iš komandų arba instrukcijų sekos, kuri vykdoma nuosekliu būdu- t. y. komanda po komandos - tradiciniuose kompjuteriuose, t. y. kompjuteriuose, kurių ideologija paremta John von Neuman pasiūlytais principais. Vis didėjantis nepatogumas, programuojant šiomis kalbomis, privedė prie to, kad buvo sukurtos naujo tipo kalbos: LISP, ISWIM, Scheme (LISP kalbos dialektas) it kt. Šio tipo kalbų sukūrimas susijęs su kalbos paprastumo koncepcijos išvystimu; jos buvo kuriamos neatsižvelgiant į konkrečius kompjuterius. Tam tikrą laiko tarpą į šią funkcionalinių kalbų klasę nebuvo kreipiamas rimtas dėmesys - tai buvo susiję su tuo, kad programos, parašytos šiomis kalbomis dirbo lėtai. Vėlesni variantai leido priartėti ar net pasiekti tradicinėmis kalbomis parašytų programų greitį klasikiniuose kompjuteriuose, netgi jeigu funkcionalinės programos nebuvo parašytos specialiai šiems kompjuteriams.

Programinės įrangos krizė

Funkcionalinių kalbų vystymuisi padėjo vis didėjantis nepatogumas, naudojant imperatyvinio tipo kalbas. Pirmą kartą apie programinės įrangos krizę buvo pradėta kalbėti 1970 metais. Programos vis sudėtingėjo ir dažnai juose būdavo gana daug klaidų, jos buvo sunkiai skaitomos, jas buvo sunku suprasti ir ypatingai sunku tobulinti. Tikėjimas, kad šios aukšto lygio imperatyvinės

kalbos supaprastins programavimą, nepasiteisino - šios kalbos nebuvo tokio aukšto lygio, tokio aukšto abstrakcinio lygio, kokio iš jų buvo galima tikėtis. Imperatyvinės kalbos iki šiol laikosi pirminių, von Neuman tipo kompjuterinių koncepcijų, ir programavimo lygis vis dar labai priklauso nuo kompjuterių išvystymo lygio. Funkcionalinės kalbos turi bruožus, įgalinančius sušvelninti kai kuriuos tradicinių kalbų apribojimus.

Papildomai informacijai ir supratimui rekomenduojamas vadovėlis: "Functional Programming Using Standart ML", Åke Wikström, Prentice Hall 1987.

Norint geriau suprasti išradimo tikslus bei privalumus ypač svarbu suprasti kas sudaro funkcionalinio metodo skaičiavimuose esmę, ypač lyginant su istoriškai labiau paplitusiu imperatyviniu priėjimu.

Išsireiškimas "funkcionalinis metodas" naudojamas norint pabrėžti, kad programos yra parašytos funkcionaline kalba ir saugomos bei vykdomos kompjuteriuose kurių elektroninė įranga specialiai pritaikyta tokio tipo kalboms. Atitinkamai, išsireiškimas "imperatyvinis metodas" naudojamas norint pabrėžti, kad programos yra parašytos imperatyvine kalba ir saugomos bei vykdomos kompjuteriuose, kurių elektroninė įranga specialiai pritaikyta imperatyvinio tipo kalboms.

Vienok, yra įmanoma saugoti ir vykdyti programas, parašyta funkcionaline kalba tradiciniuose kompjuteriuose. Atvirkštinis variantas yra taip pat įmanomas - programos parašytos imperatyvine kalba gali būti paleistos kompjuteriuose, pritaikytuose programoms, parašytoms funkcionaline kalba.

Redukuojantys procesoriai grafų minimizavimui buvo aptarti pastarųjų metų literatūroje, ir buvo pasiūlytas apdorojimo mechanizmas, aprašytas D. A. Turner "A New Implementation Technique for Applicative Languages", Software-Practice and Experience, Vol. 9, psl. 31-49, 1979. US-PS-4,644,464 aprašyta įranga realizuojanti idėjas, paskelbtas minėtame straipsnyje, kurios esmė yra procesorius, vykdantis programas, pateiktas dvejetainių kryptinių grafų pavidale, naudojant ekvivalentinių grafų paieškos mechanizmą. Gali būti naudojamos tokios kalbos, kaip LISP arba SASL. Siekiant padidinti programų vykdymo greitį US-PS-4,644,464 aprašomas taip

vadinamas grafų administratorius, kuris redukuoja kombinatorinį grafą, aprašomą nepriklausomų kintamųjų taikomosios kalbos kodais, kurie yra yra objektinių kodų rūšis, gaunamų transformuojant aukšto lygio kodus į tam tikrą skaičių registrų vienkartinio kopijavimo operacijomis. Dviejų lygių viršūnės yra aprašomi viršūnių tipo laukais. Grafas minimizuojamas, panaudojant pakeitimo funkciją, ir pertvarkant tam tikrą skaičių registrų, kuriuose gali būti saugomi dviejų ląstelių dydžio grafo viršūnės. Viršūnės, kurios yra kitų duomenų sekcijos viršūnių tėvai, kartu su tam tikru skaičiumi registrų saugomi stekinėje atmintyje ir naudojami kaip kelio buferis. Sąlygų tikrintojas valdo registrus ir tokiu būdu generuojamas pernešimo adresas mikroinstrukcijoms, kurios indikuoja kokie pertvarkymai grafe yra padaryti.

US-PS-4,734,848 aprašo redukuojantį skaičiavimo metodą ir atitikamą aparatą, kuris remiasi principais aprašytais D. A. Turner. Redukuojančio procesoriaus sistema vykdo programą, transformuotą į atitinkamą grafą, įjungiantį elementų sąrašus, funkcijas ir kitamuosius. Atitinkamai transformacija į taip vadinamą "kombinatorinį" grafą vykdoma pagal Turner sistemą, kada funkcija su daugeliu argumentų yra sutvarkoma ir transformuojama į aukštesnės eilės funkciją su vienu argumentu. Kombinatorinis grafas, gautas po apibendrinimo, tampa ištęstu, ir todėl naudojant Turner metodą, minimizavimo proceso žingsnių skaičius išauga. Todėl US-PS-4,734,848 aprašoma sistema, kuri gali naudoti lygiagretų redukavimą (minimizavimą), kartu su lėtu įvertinimu, kaip efektyvų metodą. Esminis sistemos principas yra tame, kad funkcija su daugeliu argumentų yra traktuojama kaip funkcija su vienu argumentu, kuris susietas su elementų sąrašo, turinčio pradinius argumentus, argumentais. Gauta vieno argumento funkcija yra konvertuojama. Lygiagretus įvertinimas realizuojamas standartine tvarka, kuri remiasi taip vadinamo V-kodo, randamo elementų sąrašo, įvertinimu. Vykdomas priskyrimas skirtingiems skaičiavimo įrenginiams ir tada vienetiniuose procesoriuose, sujungtuose į matricą, vykdomas redukavimas. Kiekvienas vientinis procesorius susideda iš keturių įtaisų, tokių kaip ALU, kontrolinio registro, stekinės srities ir ROM (tiktai nuskaitomos atminties), kurioje saugoma bazinės instrukcijos. Taip pat aprašomas aparatas, susidedantis iš pagrindinio procesoriaus įrenginio (PPĮ),

pagrindinės įrenginio atminties, stekinės atminties, naudojamos redukavimo stekams saugoti, ir registrinės atminties masyvo nuorodų panaudojimui pagreitinti. Naudojimas yra realizuotas mikroinstrukcijomis.

Funkcionalinės kalbos savybės

Programa, parašyta funkcionaline kalba, atrodo kaip objektų savybių apibrėžimų seka ir skaičiavimo taisyklių visuma. Apibrėžimai yra deklaratyvioji dalis, o skaičiavimo, arba redukavimo, arba perrašymo taisyklės yra vykdomoji programos dalis, kompjuterio naudojama vykdymo metu. Funkcionalinės kalbos palaiko aukšto lygio interfeisą bendravime su kompjuteriu, interfeisą, kuris leidžia programuotojui neprisirišti prie komjuterio elektroninės dalies ypatybių. Kaip dar vieną funkcionalinių programų, skirtingai negu tradicinių programų, bruožą reikia paminėti jų trumpumą ir geresnį suprantamumą. Viena iš didžiausių funkcionalinių programų blogybių yra ta, kad jos turi būti transliuojamos į tradicinę kalbą, kad galėtų būti vykdomos tradiciniuose kompjuteriuose. Tai yra daroma kompiliatorių arba interpretatorių pagalba. Yra aišku, kad kai kurie funkcionalinės kalbos privalumai yra nerealizuojami, kadangi neegzistuoja atitinkama elektroninė įranga, leidžianti efektyviai vykdyti funkcionalines programas.

IŠRADIMO OBJEKTAI

Pagrindinis išradimo tikslas yra procesoriaus sukūrimas, kurio pagalba būtų pašalinta daugelis kompjuterio elektroninės dalies apribojimų.

Kitas išradimo objektas yra yra labai greito procesoriaus, pritaikyto realaus laiko uždaviniams vykdyti, sukūrimas.

Dar kitas išradimo tikslas yra procesoriaus su kiek įmanoma paprastesniu ir unifikuotu programavimu sukūrimas.

Dar sekantis išradimo tikslas yra sukurti procesorių, kurio programavimo kalba būtų naudojama tiek kaip mašininė kalba, operacinės sistemos kalba, bendravimo protokolams palaikyti, ir kaip programavimo kalba, t. y. kalba turėtų būti paprasta ir šios kalbos lygių skaičius turėtų būti minimalus.

Kitas išradimo tikslas yra sukurti procesorių, kuris būtų labai

parastas ir lanksčiai naudojamas tiek programuotojų, tiek sistemos administratorių, kompjuterinės įrangos kūrėjų bei aptarnaujančių žmonių.

Dar kitas išradimo tikslas yra sukurti procesorių, pritaikytą deklaratyviai programavimo kalbai, opeuojančiai semantiniais elgesiniais objektais, tokiu būdu aprašant realaus laiko ir sistemos savybes.

Dar kitas išradimo tikslas yra sukurti procesorių, įjungiantį kartu dirbančius atskirus struktūrinį aritmetinį ir skaitmeninį aritmetinį procesorius.

IŠRADIMO ANOTACIJA

Atsižvelgiant į paminėtus ir kitus objektus, išradimas apima kontroliuojamą programos su tam tikra struktūra redukuojantį procesorių, kuris gali minėtą struktūrą redukuoti per tam tikrą skaičių skirtingų etapų, ir šio tipo pirmos eilės procesorius įjungia aktyvią atmintį, susidedančią iš

a. eilės aktyvios atminties ląstelių, kurių kekviena turi galimybę pakelti redukavimo efektyvumą
b. ryšio tinklo, sujungiančio visas ląsteles, ir kurio pagalba perduodami redukavimo, atlikto tose ląstelėse, rezultatai.

Ryšio tinklas įjungia magistralės sistemą, susidedančią iš valdymo ir duomenų linijų, prijungtų prie kiekvienos iš minėtų lastelių, kur valdymas suprantamas bendras visoms minėtoms ląstelėms. Kiekviena atminties ląstelė turi saugoti informaciją, reikalingą redukavimo operacijai vykdyti.

Redukavimo informacija turi įjungti mažiausiai vieną nuorodą į minėtas atminties ląsteles, kurios turinys minėtos nuorodos pagalba įjungiamas į medžio struktūrą. Mažiausiai viena iš minėtų atminties ląstelių, taip vadinamoji bazinė ląstelė, gali vykdyti visus redukavimo tipus. Objekto saugojimo atminties (objektinės atminties) ląstelės priklauso asociatyvinei atminčiai, kurios viena magistralės sistema skirta išorinei kontrolei, o kita - duomenų perdavimui. Ši atmintis įjungia:

kelias minėtas objektinės atminties ląsteles kompleksinei informacijai saugoti.

kiekviena minėta objektinės atminties ląstelė saugo mažiausiai vieną žymę, indikuojančią minėtos ląstelės išrinkimo būseną (išrinkta ar neišrinkta)

atliekamos paieškos operacijos minėtose objektinės atminties lątelėse, užstatant minėtas žymes

prijungtas prie minėtų objektinės atminties ląstelių prioritetų dekoderis, kuris išrenka vieną iš kelių minėtų objektinės atminties ląstelių.

Mažiausiai vienos bendros (globalinės) magistralės pagalba galima tarp minėtų atminties ląstelių atlikti TAIP ir ARBA rūšies loginius veiksmus, keistis informacija šios magistralės pagalba ir kontroliuoti minėtų atminties ląstelių indėlį į vykstančias logines operacijas.

Keli pirmos eilės redukuojantys procesoriai gali būti sujungti į kvadratinį lauką (matricą) tokiu būdu sudarydami antros eilės redukuojantį procesorių, kur kiekvienas pirmos eilės procesorius minėtame lauke kanalų pagalba prijungtas prie savo kaimynų.

Antros eilės redukuojantis procesorius gali būti sudalintas į logines sritis, kurių kiekviena gali būti 2^n kartų 2^n pirmos eilės procesorių dydžio. Minėtos loginės sritys tvarkingai išsidėsčiusios viena šalia kitos ir perdengia visą minėto antros eilės redukuojančio procesoriaus kvadratinį lauką.

Mažiausiai vieno nurodančiojo netiesioginio elemento, saugomo kiekvienoje srityje, adresas yra prieinamas duotos srities atminties ląstelėms.

Pirmos eilės procesorius gali būti sujungtas su kitais pirmos eilės procesoriais, sudarant antros eilės procesorių, kiekviena minėta antros eilės procesoriaus sritis gali būti perkelta per pusę srities dydžio visomis kryptimis šios srities vidinio peradresavimo antros eilės procesoriuje pagalba.

Kaip alternatyva, keli minėti pirmos eilės redukuojantys procesoriai gali būti sujungti į magistralinių arba žiedinių tinklų hierarchinę sistemą, arba keli minėti pirmos eilės redukuojantys procesoriai gali būti sujungti į mažiausiai dviejų tipų tinklų hierarchinę sistemą, iš kurių pirmasis yra magistralė, antrasis - žiedas ir trečiasis - kvadratinis laukas.

Mažiausiai viena porto reikšmė (turinys) atspindima minėtos aktyvios atminties reikšme, ir mažiausiai viena kaimyninė reikšmė atspindima mažiausiai vienoje porto reikšmėje (turinyje).

Lyginimui naudojama ateinančio į portą signalų sekos reikšmės yra įrašomos mažiausiai į vieną atminties ląstelę. Porto reikšmės susideda iš: neapibrėžto skaičiaus įrašomų elementų ir reikšmių, atspindinčių loginį neigimą, jeigu sulyginimo operacijos metu gaunamas ryškus skirtumas, ko pasekoje įrašyti elementai ištrinami iš atminties; o priešingu atveju elementai unifikuojami ir išsaugomi.

Tam tikras topologiškai atskirtų antros eilės redukuojančių procesorių gali sudaryti trečios eilės redukuojantį procesorių.

Apspręstas pagal sritis asociatyvinis adresavimas yra naudojamas antros eilės redukuojančiuose procesoriuose, o trečios eilės redukuojančiuose procesoriuose naudojamas fizinis adresavimo būdas.

Vykdymas, naudojant redukavimą (minimizavimą)

Vykdymui paruoštą programą galima apibrėžti kaip orientuotą išraiškų grafą, kur kiekviena programos dalis yra atskira išraiška. Vykdymo metu šis orientuotas išraiškų grafas yra laipsniškai redukuojamas (minimizuojamas) pagal naudojamos kalbos taisykles. Kai jau nebelieka vykdomų išraiškų, programa baigia darbą. Orientuotas išraiškų grafas gali būti atspindimas kaip medžio struktūra, kur kiekviena viršūnė yra išraiška, ir kur aukščiausia viršūnė yra vadinama šaknimi. Vykdymas, naudojant redukavimą (minimizavimą) yra atliekmas redukuojant medžio struktūra iš apačios į viršų, pirmiausia redukuojant medžio dalis, esančias toliausiai nuo šaknies ir toliau tęsiant redukavimą šaknies link. Šis redukavimo būdas yra vadinamas redukavimas pagal pareikalavimą, t. y. programos dalių, reikalaujančių tam tikrų kitų programos dalių darbo rezultatų, vykdymas pristabdomas, kol minėti rezultatai nebus prieinami.

APIBRĖŽIMAI

Žemiau pateikimas naudojamų terminų sąrašas ir jų apibrėžimai:

elementas

dalis kažko didesnio duomenų struktūroje

sąrašas

tam tiktu būdu sutvarkyta elementų eilė, kurios kiekvienas elementas gali būti savo ruožtu sąrašu

įstatytas sąrašas

pakankamai maža sąrašo dalis, kuri visa gali būti sagoma vienoje ląstelės išraiškoje. Įgalina padaryti pasirenkamus ilgus sąrašus

išraiška

hierarchiškai sutvarkyta struktūrinė visuma apibrėžianti procesą. Visos išraiškos turi šaknį, kurios pagalba identifikuojamos išraiškos. Redukavimas yra vykdomas išraiškų redukavimo mechanizmo pagalba.

objektinė atmintis

atmintis, kurios ląstelėse saugomas objektas

atminties ląstelė

ląstelė objektinėje atmintyje. Joje saugoma ląstelės išraiška, kurioje gali būti nuorodos į kitose ląstelelėse saugomas išraiškas.

ląstelės išraiška

atminties ląstelės turinys

atminties ląstelės laukas

laukas atminties ląstelėje

išraiškos elementas

duomenų elementas saugomas atminties ląstelės lauke

išraiškos identifikatorius

ląstelės išraiškos elementas identifikuojantis išraišką

kanoninė išraiška

išraiška, kuri negali daugiau būti redukuojama, t. y. išraiška, kurioje nėra nei vieno kitų, potencialiai redukuojamų išraiškų, identifikatorių.

tikslas (uždavinys)

išraiškos redukavimo rezultatas, išraiška po redukavimo

tėvas (motina)

išraiška, kurios reikšmės/nuorodos lauke yra mažiausiai vienas kitos išraiškos identifikatorius

sūnus

išraiška identifikatoriaus pagalba prijungta prie kitos išraiškos, kurioje ir nurodytas esamos išraiškos identifikatorius.

Sūnus gali būti ir tėvas. Tėvas gali būti ir sūnus. Sūnus gali turėti kelis tėvus. Tėvas gali turėti kelis sūnus.

išraiškų tinklas

išraiška ir visi jos tėvai

išraiškos vieta

nustato ar išraiška yra šaknis ar paprasta viršūnė

šaknis

aukščiausia (pirmutiniausia) išraiškų medžio išraiška

viršūnė

išraiškų medžio išraiška, kuri nėra šaknis

kur

atminties ląstelės laukas, kuriame saugoma išraiškos vieta (padėtis)

tipas

tipo kodas ląstelės išraiškoje, t. y. bitų kombinacija apsprendžianti objekto savybes, pvz. tai gali būti instrukcijos kodas

veiklumas

ląstelės išraiškos elementas, indikuojantis išraiškos stovį: - ar išraiška yra vykdoma, ar pristabdyta ar neaktyvi.

identifikatorius

ląstelės išraiškos elemento rūšis, naudojama objekto,

saugomo atminties ląstelėje, identifikavimui

aplinka, kontekstas

objektai gali būti grupuojami nustatant jiems tokį patį kontekstą

reikšmė/nuoroda

išraiškos elementas, kuriame gali būti saugoma arba tam tikra reikšmė (tame tarpe ir niekas nesaugoma), arba nuoroda į kitą išraišką.

bazinė ląstelė

struktūros aritmetinis įrenginys. Bazinė ląstelė gali aritmetiškai pertvarkyti struktūrą, įtraukdama redukuojančias išraiškas

skaitmeninis ALU

skaitmeninis aritmetinis įrenginys, galintis vykdyti bazines logines ir skaitmenines operacijas. Bazinė ląstelė naudoja skaitmeninį ALU skaitmeninėms operacijoms atlikti

bendras registras

registras, pasiekiamas visose bazinės ląstelės plokštumuose (sluoksniuose)

bazinės ląstelės žodis

bazinės ląstelės bendro registro turinys

apribotas registras

bazinės ląstelės registras, pasiekiamas tik tam tikrose plokštumuose (sluoksniuose), skirtas ląstelės išraiškos elemento įjungimui į reikšmę/nuorodą.

elemento žodis

apriboto registro turinys, arba pilno registro dalis, turintį tuos pačius išplėtimus, kaip ir apribotas registras

skaitmeninis žodis

elemento žodžio dalis su reikšme arba nuoroda

požymio žodis

elemento žodžio dalis, turinti požymį, indikuojantį skaitmeninio žodžio savybes

redukavimas (minimizavimas)

išraišų perrašymas/pertvarkymas pagal tam tikros programavimo kalbos taisykles

ASOCIATYVINĖ OBJEKTINĖ ATMINTIS

Kuriant išradime aprašytą procesorių, buvo laikomasi principo, kad visų pirma reikia sukurti objektinę atmintį iš kelių atminties ląstelių. Kiekviena tokia ląstelė gali būti laisva, arba joje gali būti įrašyta išraiška. Ląstelės nerūšiuojamos tam tikra tvarka, bet priskiriamos pasiekiamų resursų sričiai.

Labai svarbu atsiriboti nuo fizinių adresų arba fizinės padėties. Priklausybė nuo tokių parametrų anksčiau ar vėliau sukels ta pačias problemos, kokios iškilo visuose tradicinę RAM naudojančiuose įrenginiuose.

Visos objektinės atminties ląstelės komunikuoja atminties magistralės pagalba. Tai labai svarbu sumažinant kainą. Kitos magistralės, kaip pvz. kelių portų panaudojimas, padidina objektinės atminties dydį. Atminties magistralėje viena operacija įvykdoma per vieną atminties ciklą.

Redukavimo mechanizmas turi turėti struktūrą, susidedančią iš ląstelių išraiškų, kurių kiekviena turi savo identifikatorių, ir kur sujungtos identifikatorių pagalba ląstelės išraiškos suformuoja grafą. Grafas gali būti pasiekiamas ląstelių išraiškų identifikatorių adresacijos pagalba. Tokiu būdu atminties magistralė gali būti panaudota grafo viršūnių pasiekimo keliui nustatyti.

Kiekviena ląstelės išraiška turi kontekstą, kuriame apibrėžtas medžio šakininės išraiškos identifikatorius, leidžiantis nustatyti išraiškos konteksto turinį. Tokiu būdu visa medžio struktūra gali būti nustatyta vienos išraiškos pagalba vienos operacijos pagalba. Išraiškos, turinčios vienodą konteksto turinį, gali būti grupuojamos.

Visas "adresavimas" atliekamas saugomos informacijos turinio pagalba, kadangi esant asociatyviai atminčiai negalimas fizinis

adresavimas. Ląstelės išraiška, t. y. atminties ląstelės turinys, susideda iš kelių elementų, saugomų atminties ląstelės laukuose. Kiekvienas ląstelės elementas gali turėti identifikatorių ir informacinę žymę, atspindinčią reikšmę arba elemento aktyvumo būseną. Asociatyvinės paieškos metu paieškos rakto šablone gali būti naudojamas bet kuris išraiškos elementas ar jų kombinacija, atspindintys pvz. ląstelės identifikatorių, ląstelės konteksto turinį, ląstelės tipą, reikšmes, įrašytas į atminties ląstelės laukus, ar jų kombinacijos.

Išraiškos elementai gali įjungti papildomą išrinkimo bitą, nurodantį, kad šis elementas yra pasiekiamas. Tam tikros paieškos operacijos gali užstatyti šiuos atminties ląstelės laukų bitus.

Toks pasiekimo mechanizmas gali išrinkti vieną arba kelias atminties ląsteles. Viena, daugelį ląstelių liečianti operacija gali įtašyti identifikatorius daugelyje išrinktų ląstelės laukų skirtingose ląstelėse.

Centrinis valdymo įrenginys

Pgrindinių redukavimo taisyklių yra tiek daug, kad jos negali būti saugomos kiekvienoje ląstelėje. Todėl redukavimo mechanizmas yra bendras visoms ląstelės išraiškoms. Valdymo įrenginiai ir programiniai stiprintuvai prijungti prie visų atminties ląstelių, leidžia tvarkyti ilgas atminties magistralės linijas. Centrinis valdymo įrenginys sutvarko tiek laikinius signalų parametrus, tiek signalų lygį. Tokiu būdu visos atminties magistralės pagalba atliekamos komunikacijos operacijos tvarkomos centrinio kontrolės įrenginio pagalba. Centrinis kontrolės įrenginys, esantis loginių elementų masyvu, daugeliu atveju perkoduoja išorinius signalus, patenkančius iš išorinių įrenginių, ir perduoda juos valdymo magistralės pagalba visoms objektinės atminties ląstelėms. Papildomai informacijai šia tema rekomenduojama Carver Mead ir Lynn Conway knyga "Introduction to VLSI Systems", Addison Wesley Publishing Company, 1980.

Norint sudaryti sudėtinio pasiekimo kombinacijas, kiekvienoje atminties ląstelėje išsprendžiamos loginės išraiškos. Šios išraiškos yra kontroliuojamos centrinio valdymo įrenginio valdymo žodžio pagalba. Šis valdymo žodis paprastai generuojamas centriniame kontrolės įrenginyje ir apibrėžia atminties ląstelėje

saugomą informaciją. Daugeliui uždavinių labai svarbi ta redukuojančio procesoriaus savybė, kad loginės išraiškos sprendžiamos pačiose ląstelėse, kas yra žymiai efektyviau, negu kaip paprastai nuskaityti informaciją iš atmunties ir tik po to ją apdoroti.

Loginės išraiškos minimizuojamos, siekiant sumažinti saugojimo zonos dydį. Atitinkamai parenkamos ir pasiekimo funkcijos.

Tokiu būdu kontrolės įrenginys valdo objektinės atminties funkcijas. Atminties magistralė leidžia komunikuoti tarpusavyje visoms atminties ląstelėms. Tam tikrais atvejais kelias atminties ląsteles reikia nuskaityti. Tam tikras mechanizmas išrenka vieną ląstelę iš šių kelių atminties ląstelių. Išrinkimas vykdomas prioritetų dekoderio, prijungto prie visų atminties ląstelių, pagalba.

BAZINĖ LĄSTELĖ

Tam tikra aktyvi atminties ląstelė, žemiau vadinama bazine ląstele, turi aptarnauti objektinę atmintį kaip aktyvią atmintį. Bazinė ląstelė gali atlikti visas redukavimo rūšis, tuo tarpu kaip iš atminties ląstelių susidedanti objektinė atmintis gali atlikti tik kai kurias kelių redukavimo tipų dalis. Tačiau tam tikra objektinės atminties dalis gali emuliuoti bazinės ląstelės funkcijas - tačiau šiuo atveju efektyvumas krenta.

Bazinė ląstelė gali saugoti daug išraiškos lygių. Išraiškos pagrindinės instrukcijos gali būti įvykdytos bazinėje ląstelėje. Bazinė ląstelė yra pritaikyta aritmetiniam struktūrizavimui - t. y. kompjuterio programos aritmetiniam pertvarkymui. Ląstelės išraiškos dydis atitinka grafo, iššaukusio pertvarkymo operacijas, šakos dydžiui.

Skaitmeninis/Struktūrinis įrenginys

Reikia pažymėti, kad bazinė ląstelė veikia struktūriniais aritmetiniais metodais, kas reikalinga daugeliui redukavimo operacijų; ir kad skaitmenėms operacijoms atlikti ji naudoja skaitmeninį aritmetinį įrenginį (skaitmeninį ALU). Paprastai procesorius naudoja aritmetinį loginį įrenginį (ALU) atlikti tiek skaitmenines, tiek dalinai struktūrines operacijas, ir toks ALU yra nedalinamas į skirtingas funkcines dalis struktūrinėms ir

skaitmeninėms operacijoms atlikti. Naudojant tradicinį ALU reikalinga nedidelė programa struktūrinėms operacijoms, atliekamoms bazinėje ląstelėje, emuliuoti.

TRUMPAS PIEŠINIŲ APRAŠYMAS

Geriasniam čia pateikto išradimo ir jo savybių bei ypatybių supratimui toliau sekančiame aprašyme bus remiamasi piešiniais, kuriuose:

Pieš.1	išradime aprašyto pirmos eilės redukuojančio procesoriaus struktūros schema
Pieš.2	pirmos eilės redukuojančio procesoriaus (pieš. 1) atminties organizacijos schema
Pieš.3	atminties (pieš. 2) ląstelės organizacijos schema
Pieš.4A	skirtingų atminties ląstelės, galinčių saugoti ląstelės išraišką, laukų naudojimo schema
Pieš.4B-4D	bazinės ląstelės registrų schema
Pieš.4E	bazinės ląstelės registrų galimos konfigūracijos schema
Pieš.5E-5F	skirtingos bazinės ląstelės duomenų saugojimo formos
Pieš.6-8	bazinės ląstelės operacijų pavyzdžiai
Pieš.9	pirmos eilės redukuojančio procesoriaus (pieš. 1) bazinės ląstelės operacijų plokštumos blokinė diagrama
Pieš.10	bazinės ląstelės bendrojo registro ląstelės, turinčio visus galimų sujungimų tipus operacijų plokštumuje (pieš. 9), schema
Pieš.11 ir 12	du bazinės ląstelės tarpregistrinio duomenų perdavimo pavyzdžiai
Pieš.13-19	bazinės ląstelės viduje vykstančio duomenų perdavimo pavyzdžiai
Pieš.20	atminties ląstelei (pieš. 3) priklausančios bitinės ląstelės ir išorinių įrenginių, prijungtų prie bitinės ląstelės, grandinės schema
Pieš.21A	į atmintį įjungto prioritetų dekoderio bloko

	grandinės schema
Pieš.21B	bloko iš pieš. 21A sujungimų schema
Pieš.22	atminties elemento aprašymo įrangos grandinės schema
Pieš.23	išraiškos elemento aprašymo įrangos grandinės schema
Pieš.24A	pirmos eilės redukuojančio procesoriaus (pieš. 1) skaitmeninio ALU schema
Pieš.24B	skaitmeninio ALU (pieš. 24A) taktinių ir kelių pagrindinių valdymo signalų diagrama
Pieš.25	bitinio lauko sudalinimo aritmetinio atvaizdavimo schematinis piešinys
Pieš.26A	sveikų skaičių kodo grafinis atvaizdavimas
Pieš.26B	plaukiojančio kablelio skaičių kodo grafinis atvaizdavimas
Pieš.26C	arimetinio atvaizdavimo 7 bitų magistralėje schema
Pieš.27A-27D	pieš. 26 naudojamas duomenų sąrašąs
Pieš.28	pirmos eilės redukuojančio procesoriaus skaitmeninio ALU grandinės schema
Pieš.29A-29C	skaitmeninio ALU išėjimo/įėjimo skirtingais laiko momentais buferio stovio schema
Pieš.30	tikslumo dekoderio, įtraukto į pieš. 28 parodytą grandinę, grandinės diagrama
Pieš.31A	viso žodžio sumatoriaus, 28 parodytą grandinę, grandinės diagrama
Pieš.31B	sumatoriaus (pieš. 31A) tikslumo dalies blokinė diagrama
Pieš.31C	tikslumo dalies (pieš. 31B) bitinės dalies blokinė diagrama
Pieš.32	grandinės, parodytos pieš. 28, inkrementatoriaus tikslumo dalies schema
Pieš.33	grandinės, parodytos pieš. 28, operandų grandinių bazinio selektoriaus bazinė bitinė dalis.
Pieš.34	procesoriaus porto signalų sekos diagrama

Pieš.35 procesoriaus įėjimo arba/ir išėjimo porto įrangos schema

Pieš.36 bendra antros eilės redukuojančio procesoriaus (SOP) schema

Pieš.37A SOP susidedančių iš pirmos eilės redukuojančių procesorių (FOP), formuojančių kvadratinį lauką, schema

Pieš.37B bazinės ląstelės keturių duomenų perdavimo tipų, naudojamų SOP (pieš. 37A) kvadratiniame lauke, operacijų plokštuma

Pieš.37C SOP, susidedančio iš 256 FOP, kvadratinio lauko, padalinto į stritis, schema

Pieš.37D-37E skirtingų programiškai formuojamų žiedų SOP lauke pavyzdžiai

Pieš.38 magistralinio tinklo, suformuoto iš FOP, sudarančių SOP, schema

Pieš.39 žiedinio tinklo, suformuoto iš FOP, sudarančių SOP, schema

Pieš.40 SOP kombinacijos, susidedančios iš pieš. 37A, 38 ir 39 SOP, schema

Pieš.41 trečios eilės redukuojančio procesoriaus, susidedančio iš kelių SOP, schema

Pieš.42A-42C skirtingų trečios eilės redukuojančio procesoriaus dalių schema, bei adresacijos pavyzdžiai

ĮRANGOS APRAŠYMAS

Pirmos eilės redukuojantis procesorius

Pieš. 1 parodytas išradime aprašytas pirmos eilės redukuojantis procesorius. Pirmos eilės procesorius yra redukuojantis procesorius, kuris gali būti panaudotas kaip toliau aprašyto labiau sudėtingesnio redukuojančio procesoriaus dalis. Redukuojančio procesoriaus idėja yra aiški iš pieš. 1. Vienok reikia pažymėti, kad elementų išdėstymas, formuojant pirmos eilės redukuojantį procesorių, gali būti skirtingas, pvz. skirtingos procesoriaus

plokštumos gali būti viena šalia kitos mikroschemoje, arba lygiagrečios plokštumos nebūtinai gali atitikti schemoje parodytą išdėstymą.

Pirmos eilės redukuojantis procesorius (pieš. 1) įjungia: aktyvią asociatyvinę atmintį 1, žemiau vadinamą objektine atmintimi (atmintimi, kur saugomas objektas), kuri sudaryta iš atminties ląstelių, galinčių saugoti įjungiantį vykdomas dalis duomenų objektą, bei galinčių atlikti ribotą redukavimo operacijų skaičių; struktūrinį aritmetinį įrenginį 2, žemiau vadinamą bazine ląstele, turinčią kelis struktūrinėms aritmetinėms operacijoms naudojamus bazinius registrus 3; ir, jeigu pirmos eilės redukuojantis procesorius iš pieš. 1 yra aukštesnės eilės redukuojančio procesoriaus dalis, įjungia tinklines duomenų perdavimo priemones 4, komunikacijai su kitais procesoriais palaikyti. Ryšio priemonės 4 įjungia kelis registrus, kuriose gali būti laikinai saugoma atminties ląstelių struktūra, kuri turi būti siunčiama kitam pirmos eilės procesoriui.

Yra galima objektinėje atmintyje rezervuoti lauką bazinės ląstelės funkcijoms emuliuoti. Bazinė ląstelė turi keletą savybių, kurių pagalba jos registrų turinys sukeičiamas ir persiunčiamas vienos operacijos metu. Emuliuojantis bazinę ląstelę atminties laukas tam sugaišta kelių operacijų laiką.

Skaitmeninis aritmetinis loginis įrenginys (skaitmeninis ALU) 5 yra prijungtas prie bazinės ląstelės 2, ir tokiu būdu, kai reikia atlikti skaitmenines operacijas, duomenys iš bazinės ląstelės yra siunčiami į skaitmeninį ALU. Tokiu būdu struktūrinės aritmetinės ir skaitmeninės aritmetinės operacijos atliekamos skirtinguose įrenginiuose.

Procesorius taip pat įjungia centrinį valdymo įrenginį, kurio pagalba valdoma: objektinės atminties 1 elementai, bazinė ląstelė 2 ir skaitmeninis ALU 5. Be to procesorius gali turėti vieną ar kelis portus ryšiui su išore palaikyti.

Centrinis valdymo įrenginys

Centrinis valdymo įrenginys 6 yra pakankamai sudėtinga loginių elementų visuma. Jo įranga neparodyta, kadangi jis yra programiškai konfigūruojamas, atsižvelgiant į infomacinių signalų pobūdį ir

turinį. Loginių elementų visuma sudaro bitų kaukės piešinį komjuterio konfigūracijai nustatyti. Tokiu būdu šios grandinės negalima detaliai parodyti, ir bus aprašomi tik ateinantys ir išeinantys signalai.

Objektinė atmintis (atmintis, kurioje saugojamas objektas)

Objektinė atmintis 1 yra žymiai "inteligentiškesnė", negu paprasta RAM tipo atmintis. Ji yra asociatyvinė, kadangi leidžia daugiau operacijų negu paprastas nuskaitymas ar įrašymas, kaip paprastoje RAM atmintyje. Atminties ląstelės bitinė struktūra parodyta pieš. 20.

Bitinė atminties ląstelė gali atlikti daugelį funkcijų, netgi jeigu ji turi tik keturias jungtis, iš kurių trys yra kontroliuojamos. Ji susideda iš nedaugelio elementų. Tai leidžia didelį skaičių įjungiantį atminties įrenginį padaryti pakankamai kompaktišką.

Objektinė atmintis yra sudalinta į atminties ląsteles, kurių kiekviena įjungia kelis išraiškos elementus, ir kurios atlieka daug funkcijų. Pvz. yra galima surasti visus duomenų elemento pasikartojimus atminties ląstelėse ir pertvarkyti šiuos surastus elementus vienu metu, panaudojus tik vieną atminties instrukciją. Kadangi atmintis yra asociatyvinio pobūdžio, ši pertvarkymo operacija užims tik du atminties ciklus, nežiūrint kiek atminties ląstelių dalyvauja operacijoje.

Pagal pieš. 2 asociatyvinė objektinė atmintis 1 susideda iš atminties ląstelių 10, sudarančių eilutes, plokštumos. Tokiu būdu atminties ląstelės atspindimos kaip eilučių stekas. Jos prijungtos prie atminties vertikalios magistralės transformuojančio interfeiso 7, žemiau aprašyto pagal pieš. 20, valdymo įrenginio ir reikšmių stiprintuvo t_1, t_2, id, env, v_0, v_1, v_2, v_3 linijų. Atminties ląstelės taip pat prijungtos prie prioritetų dekoderio 11, išrenkančio vieną iš kelių ląstelių operacijos vykdymui. Objektinė atmintis yra kontroliuojama suformuojamo per kelių ciklų laikotarpį asociatyvinio pasiekimo mechanizmo pagalba. Visiškai objektinė atmintis pasiekiama vieno laiko vieneto metu.

Portas

Ryšiams su išore palaikyti procesorius gali naudoti vieną ar

kelis portus, skirtus duomenų įvedimui/išvedimui. Portą galima laikyti atminties dalimi, kadangi jo atminties ląstelės vykdo panašias funkcijas kaip objektinės atminties ląstelės. Portas vykdo įvedimo/išvedimo operacijas kartu normalizuodamas (unifikuodamas).

Objektinės atminties ląstelė

Objektinės atminties 10 ląstelė saugo skaitmeninę informaciją ir kartu aktyviai dalyvauja skaičiavimuose. Sudėtinė skaitmeninė informacija ir mažiausiai viena žymė saugoma kiekvienoje atminties ląstelėje 10. Žymės gali būti CHOSEN (IŠRINKTA) arba NON CHOSEN (NEIŠRINKATA). Sudėtinė skaitmeninė informacija atminties ląstelėje, arba jos dalyse, vadinamose išraiškos elementais, gali būti nuskaitoma arba įrašoma nuskaitymo ar įrašymo operacijų metu, kai jos pažymėtos CHOSEN. Pasiekti šią informaciją galima ir ignoruojant šios žymės bito reikšmę. Tada pasiekimo mechanizmas realizuojamas kaip vieno bito magistralių a ir b , prijungtų prie atminties ląstelių (pieš. 3), loginio sprendimo rezultatas.

Dviejų bendrų vieno bito magistralių 12 (MODE - REŽIMAS) ir 13 (MORE - DAUGIAU) pagalba atliekamos loginės operacijos tarp atminties ląstelių. Trečioji magistralė 14 (ANY - BET KURIS) prijungta prie prioritetų dekoderio. Vienok magistralių skaičius nėra apribotas trimis - jų gali būti viena arba kelios. Atskiros atminties ląstelės magistralių 12, 13, 14 pagalba gali dalyvauti loginėse operacijose. Magistalė 12 (MODE) įgauna "TAIP" reikšmę, kai naudojamos magistralės a arba b. Visos atminties ląstelės gali rašyti ir nuskaityti magistralės 12 (MODE) reikšmes. Magistalė 13 (MORE) įgauna "TAIP" reikšmę, kai išrenkama daugiau kaip viena atminties ląstelė. Magistalė 14 (ANY) įgauna "TAIP" reikšmę, kai bet kuri ląstelė pareikalauja ryšio.

Atminties ląstelė, parodyta pieš. 3, padalinta į kelis laukus 151. Kiekvienas atminties ląstelės laukas 151 įjungia kelias bitų ląsteles 15 ir elemento viršūnę 166. Išraiškos elementas yra saugomas bitų ląstelėse kartu su žyme CHOSEN arba NOT CHOSEN, kuri saugoma elemento viršūnėje 16. Išraiškos elementai gali būti priskirti skirtingiems uždaviniams. Saugomo kiekviename išraiškos elemente žodžio ilgis gali būti pvz. 38 bitai. Papildomai prie žymės, saugomos elemento viršūnėje 16, keli bitai bitų ląstelėje 15 gali būti naudojami kaip aprašantys saugomą informaciją požymiai.

Taip vadinami požymių žodžiai gali būti saugomi šiuose bituose. 6 bitai atminties ląstelėje 15 naudojami požymių žodžiui, o likę 32 - paprastam saugojimui.

Kaip matyti iš pieš. 3 išraiškos elementai yra nevienodo dydžio. Taip, išraiškos elementai, prijungti prie magistralės linijų t1 ir t2, t. y. TYPE (TIPAS) yra mažesni negu elemrntai prijungti prie kitų magistralės linijų. Atminteies ląstelės laukų 151 viršūnė 16 prijungta prie vidinių vieno bito magistralių a ir b. Šių magistralių skaičius gali įvairuoti, svarbu kad būtų ne mažiau kaip viena magistralė. PRIJUNGTAS TAIP ir PRIJUNGTAS ARBA tipo loginės operacijos gali atlikti CHOSEN išraiškos elementai, panaudami šias magistrales.

Kiekvienos atminties ląstelės išraiškos viršūnė 17 prijungta prie magistralių a ir b. Išraiškos viršūnė 17 taip pat prijungta mažiausiai vieno bito pločio magistralės pagalba prie prioritetų dekoderio, kuris duotu atveju turi dvi magistrales 18 ir 19, ir taip pat prijungta prie prie bendrų magistralių 12 ir 13. Išraiškos viršūnė 17 naudojama kaip buferis. Ląstelės gali skaityti šių magistralių reikšmes arba dalyvauti loginėse operacijose.

Objektinės atminties ląstelių panaudojimas

Atminties ląstelės įranga, parodyta pieš. 4A, bus panaudota aiškinant skirtingų atminties laukų funkcijas. Atminties ląstelės laukai pieš. 4A yra ne tokie patys, kaip laukai pieš. 3, kadangi pieš. 3 parodyta aparatūrinis realizavimas, o pieš. 4A - ląstelių panaudojimas. Kaip matosi iš pieš. atminties ląstelė gali saugoti dviejų tipų išraiškos elementus ir turi laukus atitinkamai adaptuotus šių elementų saugojimui. Šių laukų vardai pieš. 4A sutampa su išraiškos elementų, saugomų šiuose laukuose, vardais.

Pirmas išraiškos elementų tipas

Pirmas elemento tipas aprašo skirtingas atminties ląstelės būsenas. Vienas iš šių elementų yra LAZY (AKTYVUMAS), žymintis ar ląstelė yra naaktyvi - tada ląstelės turinyas interpretuojamas kaip pasyvi informacija, ar ląstelė yra aktyvi, ar laukimo būsenoje - t. y. ląstelės veikimas pristabdytas ir ji laukia tam tikros informacijos, kad gelėtų tęsti savo darbą. LAZY laukas susideda iš dviejų bitų. Kitas elemento tipas yra WHERE (KUR), nurodantis ar

išraiška yra grafo šaknyje ar tai paprasta viršūnė. Šis tipas užima vieną bitą. Sekantis elemento tipas yra TYPE, pažymintis tipo kodą (par, seq, list, unify ir t. t.). Jo ilgis - trys bitai. Šie elementų tipai saugomi kaip bazinės ląstelės registrų dalys, veikiančios atitinkamose plokštumuose LAZY. WHERE ir TYPE kai ląstelės išraiška perkeliama į bazinę ląstelę redukavimui įvykdyti.

Bazinės ląstelės papildomos plokštumos CLOS/SIMPLE (IŠRAIŠKA/REIKŠMĖ) pagalba nustatomos registro reikšmės, indikuojančios išraiškos ar reikšmės buvimą. Plokštumos LAZY, WHERE, TYPE ir CLOS/SIMPLE toliau vadinamos ATTRIBUTE (ATRIBUTŲ) plokštumomis. Priklausomai nuo vykdomo uždavinio tipo gali būti realizuota ir daugiau atminties ląstelės laukų bei bazinės ląstelės plokštumų.

Antrasis išraiškos elementų tipas

Antrasis elementų tipas nusako identifikatorių, kontekstą arba reikšmę - tai IDENTIFIER (IDENTIKATORIUS), ENVIROMENT (KONTEKSTAS) ir VALUE/DES (REIKŠMĖ/NUORODA). Šie elementai saugomi bazinės ląstelės 3 registrų dalyse ir yra pasiekiami plokštumose HEAD (VIRŠŪNĖ) ir NUM (SKAITMENINIS), parodytose pieš. 1. Kiekvienas iš šių elementų turi 38 bitų žodį, kuris padalintas į dvi dalis, iš kurių viena , 32 bitų ilgio, saugoma NUM plokštumoje, o kita 6 bitų ilgio dalis saugoma HEAD plokštumoje, kai ląstelės turinys pernešamas į bazinę ląstelę. Gali būti realizuota ir daugiau papildomų plokštumų.

Požymių žodis

Kiekvienas antrojo tipo išraiškos elementas turi požymių žodį, nustatantį skaitmeninio žodžio savybes. Šie požymiai gali būti dviejų rūšių: netiesioginiai požymiai, t. y. požymiai, naudojami identifikatoriams ir kontekstui pažymėti; ir tiesioginiai požymiai, t. y. požymiai, naudojami paprastoms reikšmėms pažymėti. Netiesioginių požymių pavyzdžiu gali būti cls, canon ir open. Jeigu požymių žodis yra cls, tai reiškia, kad skaitmeniniame žodyje randasi išraiška, kurią galima redukuoti. Jeigu požymių žodis yra canon, tai reiškia, kad skaitmeniniame žodyje randasi išraiška, jau nebegalima toliau redukuoti. Jeigu požymių žodis yra open, tai reiškia, kad identifikatoriaus lauke randasi išraiška, kuri yra įstatytas sąrašas. Tiesioginių požymių pavyzdžiu gali būti discr,

cont, unused ir nothing. Jeigu požymių žodis yra discr, tai reiškia, kad skaitmeniniame žodyje yra sveikas skaičius. Jeigu požymių žodis yra cont, tai reiškia, kad skaitmeniniame žodyje yra plaukiojančio kablelio skaičius. Jeigu požymių žodis yra unused , tai reiškia, kad skaitmeninis žodis yra nepibrėžtas. Jeigu požymių žodis yra nothing, tai reiškia, kad skaitmeniniame žodyje nieko nėra, taip pvz. jeigu išraiškoje yra laukas, pažymėtas šiuo požymiu, tai išraiška irgi nieko nieko nereiškia. Gali būti realizuota ir daugiau papildomų tiesioginių ar netiesioginių požymių žodžių.

Identifikatoriai

Jeigu atminties ląstelės identifikatoriaus laukas įjungia identifikatoriaus elementą, atminties ląstelės išraiška turi būti perkelta į bazinę ląstelę. Kiekvienos atminties ląstelės laukas VALUE/DES gali turėti identifikatorių aprašantį kitos ląstelės išraišką, tokiu būdu realizuojant ryšį su kitos ląstelės išraiška. Saugomas išraiškas galima įsivaizduoti kaip orientuotą grafą ar medį, kurio viršūnėse randasi identifikatoriai.

Kontekstas

Konteksto lauke yra saugomas šaknies išraiškos identifikatorius, kurio pagalba nustatoma duotos išraiškos vieta išraiškų medyje. Tokiu būdu vienos operacijos pagalba galima nustatyti viso medžio struktūrą. Konteksto laukas gali būti panaudotas ir kitiems tikslams. Jame galima išsaugoti sukūrusio išraišką objekto identifikatorius. Kitu atveju visos išraiškos su tais pačiais vardais ir vienodais kontekstais gali būti grupuojamos.

Jeigu išraiškos kontekstas yra apibrėžtas, pagal jį galima surasti šakninę išraišką. Šakninės išraiškos kontekstas lauke WHERE žymimas tam tikra žyme (pvz. "1"), paprastos išraiškos - skirtinga žyme (pvz"0").

Tipas

Atminties ląstelės lauko TYPE turinys žymi ląstelės išraiškos tipą. Jis gali būti, pvz.: par, t. y. lygiagrečios reikšmės ar/ir nuorodos; list, t. y. reikšmių ar/ir nuorodų sąrašas; unify, t. y. reikšmių ar/ir nuorodų turinys kuris turi būti patikrintas; ir t .t.

Bazinė ląstelė

Bazinė ląstelė yra ištisas aritmetinis įrenginys, skirtas struktūrinėms aritmetinėms operacijoms vykdyti, turintis kelis bazinius registrus 3 (pieš. 1). Registruose saugomas sąrašų medis, kur kekvienas sąrašas susideda iš žodžių

Baziniuose registruose saugoma informacija gauta iš objektinės atminties 1. Magistralės 8, įjungiančios perdavimo linijas type, id, env, v_0, v_1, v_2, v_3, bei pakankamai plačios, kad galėtų realizuoti ryšį, pagalba baziniai registrai yra prijungti prie objektinės atminties. Bazinėje ląstelėje gali būti naudojama trijų lygių objektinė struktūra. Galima išskirti tris atvejus: kai 0, 1, 2 ir 3 lygio objektų struktūra gali būti saugoma bazinėje ląstelėje.

Bazinės ląstelės registrai

Bazinės ląstelės registrų schema parodyta pieš. 4B-4D, jų konfigūracija parodyta pieš. 4E.

Pieš. 4B parodytas registras sudarytas iš registro ląstelių, kurių kiekviena saugo vieną informacijos bitą. Registrai pasiekiami skirtingose bazinės ląstelės plokštumose, ir kiekvienas registras naudoja skirtingą plokštumą.

Pieš. 4C parodytas pilnas registras, t. y. registras, kuris naudojamas visose plokštumose. Tokio tipo registras gali saugoti identifikatorių arba reikšmę savo ląstelėse, prijungtose prie NUM ir HEAD plokštumų. Jame taip pat gali būti saugoma aprašyta anksčiau būsena. Ji saugoma registro ląstelėse prijungtose BOOL, TYPE, WHERE, LAZY ir CLOS/SAMPLE plokštumose.

Pieš. 4D parodytas apribotas registras, t. y. registras tik tam tikrose plokštumose, pvz. NUM ir HEAD.

Pieš. 4E parodyta galima bazinės ląstelės registrų konfigūracija. Bazinės ląstelės registrai sujungti į kvadratą, vadinama bazine registrų matrica. Baziniai registrai turi viename savo šone eilutę, vadinamą pagrindiniais registrais. Bazinių registrų kolonėlė, viršuje turinti pagrindinį registrą yra vadinama papildomais registrais. Bazinėje ląstelėje taip yra identifikatoriaus ir konteksto registrai. Viena papildomų registrų eilutė yra išdėstyta šalia bazinių registrų matricos.

Visi bazinės ląstelės baziniai registrai, išskyrus pagrindinius registrus, gali būti tokio tipo, koks parodytas pieš. 4D, t. y. apriboti registrai, visi kiti registrai gali būti tokio tipo, koks parodytas pieš. 4C, t. y. pilni registrai.

Prieš pradedant smulkų bazinės ląstelės aparatūrinės įrangos aprašymą, bus pateikta trumpas skirtingų saugojimo formų aprašymas pagal pieš. 5A-5F ir jų panaudojimo pavyzdžiai pagal pieš. 6A-6H, 7A-7G ir 8A-8G.

Paprastos reikšmės

Kaip parodyta pieš. 5A, redukavimo rezultato reikšmė 25 saugoma atitinkame pagrindiniame registre.

Vieno lygio struktūra

Redukavimui skirtas uždavinys pakraunamas į bazinę ląstelę. Kaip parodyta pieš. 5B, vieno lygio uždavinys, kuris paprastai yra išraiška be nuorodų į kitų ląstelių išraiškas, yra įrašomas pagrindiniuose registruose. Pavyzdyje parodyta paprasta skaitmeninė operacija, t. y. reikšmių 1, 2 ir 3 sudėjimas. Skaitmeninė instrukcija (+) saugoma pirmajame pagrindiniame registre, o operandai - kituose pagrindiniuose registruose.

Dviejų lygių struktūra

Kaip matyti iš pieš. 5C, dviejų lygių struktūros medis turi savo šakninį sąrašą, kuris yra tėvas, saugomą pagrindiniuose registruose ir sūninius sąrašus, saugomus baziniuose registruose. Šiame pavyzdyje, struktūra atspindima sąrašu ((1 2) (3 4)) yra saugoms bazinių registrų matricoje. Šakninis sąrašas, t. y. 1 ir 3 saugomas pagrindiniuose registruose, o sūniniai sąrašai, t. y. (1 2) ir (3 4), yra saugomi papildomuose registruose. Kiti šio saugojimo pavyzdžiai bus pateikti vėliau, t . y. pagal pieš.6.

Trijų lygių struktūra

Kaip matyti iš pieš. 5E, uždavinio medžio, turintčio trijų lygių struktūrą, šaknis saugoma viename iš papildomų registrų ir jos vienas sūnus saugomas pagrindiniuose registruose. Pieš. 5D uždavinio medžio šaknis, kuri yra transponavimo instrukcija (Tr), saugoma viename iš papildomų registrų ir jos vienas sūnus, kuris yra sąrašas (id1, id2, id7), saugomas pagrindiniuose registruose.

Kiekvienas sąrašo elementas yra sūnaus identifikatorius. Pieš. 5E tie sūnūs yra vertikaliai pakrauti į bazinius registrus, kur id1 pakeičiamas savo reikšme, t. y. (1, 2, 3), id2 pakeičiamas savo reikšme, t. y. (11, 12, 13) ir id7 pakeičiamas savo reikšme, t. y. (21, 22, 23).

Konvejerinis režimas

Kaip matyti iš pieš. 5F, medžio, saugomo konvejerio režime, uždavinio sąrašas pakraunamas į pagrindinius registrus ir uždavinio tėvas pakraunamas į papildomus registrus, ir šis pakrautas medis turi instrukcijas ir elementus saugomus abiejų tipų registruose. Konvejerinis operacijų režimas naudojamas redukuojant skaitmenines išraiškas. Vienas iš šio režimo privalumų yra tas, kad tarpinės reikšmės gali būti saugomos ne objektinėje atmintyje, o pačioje bazinėje ląstelėje.

PAVYZDYS 1

Pirmajame pavyzdyje (pieš. 6A-6H) yra parodytas redukuojamos išraiškos lygiagrečių verčių unifikavimas

unify(par(1 par(1) 3) par(1par(1) 2)

Ši redukuojama išraiška turi būti pakeista į unifikuotą lygiagrečią struktūrą.

Pieš. 6A parodyta pradinė redukuojama išraiška. Pieš 6B parodyta kaip ši išraiška saugoma objektinėje atmintyje. Atminties ląstelės, kuriose saugomos išraiškos dalys yra pažymėtos (pieš. 6A). Ryšiai tarp elementų išraiškų ir ląstelės išraiškų yra pažymėti pieš. 6B. Ląstelės išraiška su identifikatoriumi id_1 turi žymę cls ir kodo tipą unify savo tipo lauke, o ląstelių išraiškos su identifikatoriais id_2, id_3 ir id_4 turi kodo tipą par savo tipo laukuose. Ląstelės išraiškos su identifikatoriumi id_1 pirmieji du reikšmė/nuoroda išraiškos elementai nurodo į ląstelių išraiškas su identifikatoriais id_2 ir id_4. Šios ląstelių išraiškos pažymėtos canon. Ląstelės išraiškos su identifikatoriumi id_2 pirmasis ir trečiasis reikšmė/nuoroda išraiškos elementai yra diskretinės reikšmės su požymiu discr, o antrasis reikšmė/nuoroda išraiškos elementas nurodo į ląstelės išraišką su identifikatoriumi id_3 ir pažymėtas canon. Ląstelės išraiškos su identifikatoriumi id_2 pirmasis reikšmė/nuoroda išraiškos elementas yra sveikas skaičius

ir pažymėtas discr. Ląstelės išraiškos su identifikatoriumi id_4 pirmasis ir trečiasis reikšmė/nuoroda išraiškos elementai yra diskretinės reikšmės su požymiu discr, o antrasis reikšmė/nuoroda išraiškos elementas nurodo į ląstelės išraišką su identifikatoriumi id_3 ir pažymėtas canon.

Kaip parodyta pieš. 6C, atminties ląstelės, turinčios ląstelės išraišką su identifikatoriumi id_1, turinys pirmiausia pakraunamas į bazinę ląstelę, įrašant identifikatorių į identifikatoriaus registrą kaip id_1, įjungiant išraiškos tipo kodą unify, ir pakraunant reikšmė/nuoroda elementus kaip uždavinį į pagrindinius registrus. Tai yra pirmasis operacijos etapas. Kaip tai yra realizuota, parodyta pieš. 13 ir aprašyta žemiau.

Kaip parodyta pieš. 6D, sūnūs su identifikatoriais id_2 ir id_4 pakraunami vertikaliai į bazinius registrus; jų pirmojo reikšmė/nuoroda elemento turinys kartu su žyme pakraunamas į pagrindinius registrus, o likę jų reikšmė/nuoroda elementai - į vertkaliai išdėstytus registrus. Kodo tipas par pakraunamas į pagrindinį registrą. Kodo tipas pakrautas į pagrindinį registrą, pasirodo TYPE plokštumoje.

Kaip parodyta pieš. 6E, bazinių registrų turinys yra transponuojamas, ir pirmas vertikalus bazinių registrų stulpelis patalpinamas pagrindiniuose registruose, o sekantis bazinių registrų stulpelis atsiduria eilutėje po pagrindiniais registrais. Transponavimo operacija parodyta pieš. 18 ir aprašyta žemiau. Kodo tipai par ir unify identifikatorių ir pagrindiniuose registruose sukeičiami valdymo įrenginio pagalba. Dabar baziniai registrai turi tėvą su trimis sūnumis, patalpintais stulpeliuose. Kiekvienas sūnus pakraunamas atgal į objektinę atmintį, panaudojant instrukciją make. Objektinė atmintis atsako, pakraudama šių sūnų identifikatorius į pagrindinius registrus. Valdymo įrenginys tvarkos CLOS/SIMPLE ir TYPE plokštumų darbą, perduodamas atitinkamų registrų reikšmes ir instrukcijas. Sūnus pervadinami nuoseklia tvarka pagal identifikatorių id_1 ir seni vardai nenaudojami.

Kaip parodyta pieš. 6F, pirmasis sūnus gauna identifikatorių id_2, sekantis sūnus, turintis išraišką su id_3, gauna identifikatorių id_4, ir trečias - id_5. Tėvas, turintis elementų išraiškas su nuorodomis į id_2, id_4, id_5, išlaiko savo identifikatorių id_1 ir

įrašomas į objektinę atmintį.

Pieš. 6G parodytos atminties ląstelės, saugančios redukuojamą išraišką

par(unify(1 1) unify(par(1) par(1)) unify(2 3))

Pati redukuojama išraiška parodyta pieš. 6H.

Pieš. 6G parodytos ląstelių išraiškos su kodo tipu unify ir LAZY lauke pažymėtos exec, ir ląstelės išraiška su identifikatoriumi id1 pažymėta wait, kas reiškia, kad visų pirma turi būti visos ląstelių išraiškos su exec turi būti įvykdytos, ir tik po to turi būti vykdoma ląstelės išraiška su identifikatoriumi id_1. Pieš. 6H išraiška gali būti po tam tikro laiko pakrauta į bazinę ląstelę tolimesniam skaičiavimui. Pvz. ląstelės išraiška su identifikatoriumi id_2 gali turėti reikšmę 1, kadangi jos reikšmė/nuoroda elementai, būdami 1 ir 1, yra vienodi, o ląstelės išraiška su identifikatoriumi id_5 duos rezultate nothing, kadangi jos reikšmė/nuoroda elementai 2 ir 3 yra nevienodi. Kieviena unfikacija atliekama skaitmeninio ALU, lyginančio reikšmes ir siunčiančio rezultatą į valdymo įrenginį 6, pagalba. Valdymo įrenginys perduoda informaciją loginių reikšmių pavidalu bazinės ląstelės pirmajam pagrindiniam registrui. Kai redukavimas baigiamas, rezultate gavus arba kanoninę išraišką, arba reikšmę, arba nieko, rezultatas yra siunčiamas į atminties laukus, ir kiekviena netiesioginė redukuojamos išraiškos nuoroda yra pakeičiama redukavimo rezultatu. Tai yra padaroma unify_id operacijos, aprašytos toliau (pieš. 16), pagalba.

PAVYZDYS 2

Šis pavyzdys demonstruoja sąrašų išplėtimo instrukcijos darbą, kai ląstelės išraiška turi įstatytus sąrašus. Šio tipo instrukcijos panaudojimas yra papildomas redukavimo etapas. Smulkesnė informacija bus pateikta žemiau pagal pieš. 19.

Galima atlikti kopijuojančio tipo redukavimo operacijas, naudojančias ex.type instrukciją ir dirbančias su išraiškomis, kaip pvz.

ex.type(1 list(2 3 list(4 5 6))7)

Išraiškos forma parodyta pieš. 7A ir jos ląstelės išraiškos -

pieš. 7B.

Kaip parodyta pieš. 7C, ląstelės išraiška su identifikatoriumi id1 pakraunama į bazinės ląstelės pagrindinius registrus kartu pakraunant identifikatorių ir kodo tipą į identifikatoriaus registrą. Kadangi antrojo pagrindinio registro turinys pažymėtas netiesioginiu elementu open, ląstelės išraiška esanti nuorodoje, pakraunama vertikaliai į bazinius registrus kaip sūnus (pieš. 7D).

Aparatūrinė instrukcija sąrašų išplėtimas, smulkiai parodyta pieš. 19, perneša diskretinį dydį 7 į trečią pagrindinį registrą, į poziciją šalia id4 trečiame baziniame stulpelyje, ir perneša sąrašo dalį antrame stulpelyje virš antrojo pagrindinio registro į trečią stulpelį, patalpindama jo mažiausią elementą (reikšmę 3) trečiame pagrindiniame registre ir nustatydama jam kodo tipą list. Kadangi antrojo pagrindinio registro turinys yra diskretinis dydis, jis pažymimas discr.

Naujas sąrašo išplėtimas gaunamas perkeliant žemiau pagrindinio registro esančio trečio stulpelio turinį į ketvirtą stulpelį, ir pažymint jį list. Trečiojo pagrindinio registro turinyje yra diskretinis dydis pažymėtas discr (pieš. 7F).

Po to ketvirto stulpelio sąrašas išsaugomas objektinėje atmintyje instrukcijos make pagalba. Jis išsaugomas atminties ląstelėje su identifikatoriumi id_2, kadangi šis jau nenaudojamas, ir identifikatoriaus id_2 atspindimas turinys siunčiamas atgal į bazinės ląstelės ketvirtąjį pagrindinį registrą (pieš. 7G).

Toliau įvykdomi kitos ex.type instrukcijos redukavimo operacijos, ir rezultatas įrašomas į objektinę atmintį.

PAVYZDYS 3

Skaitmeninė instrukcija, kuri gali būti +, -, *, /, mod ir t. t., turi būti įvykdyta. Po instrukcijos seka jos argumentai. Šiame pavyzdyje turi būti sudėti sąraše esantys skaičiai. Redukavimo operacija atliekama išraiškai

apply(+ list(1 2))

Uždavinys parodytas pieš. 8A, o jo ląstelės išraiškos - pieš. 8B.

Kaip parodyta pieš. 8C, ląstelės išraiška su identifikatoriumi id_1 pakraunama į bazinės ląstelės pagrindinius registrus, įrašant

identifikatorių ir kodo tipą į identifikatoriaus registrą. Skaitmeninė instrukcija (+) yra pažymėta kaip instrukcija. Kadangi antrojo pagrindinio registro turinys pažymėtas kaip netiesioginis elementas open, surištos ląstelės išraiška yra veritkaliai pakraunama į bazinius registrus (8D).

Naujas sąrašo išplėtimas gaunamas, pažymint diskretinį dydį antrajame pagrindiniame registre kaip discr bei sąrašą 2 kaip list. Tokios pačios operacijos yra atliekamos ir kai identifikatoriaus id_2 sąrašas susideda iš dviejų, trijų ar keturių elementų. Kadngi šiuo atveju sąraše yra tik vienas elementas, žymė list pakeičiama žyme discr, tokiu būdu nurodant kad pagrindiniame registre yra diskretinis dydis (pieš. 8F).

Kadangi pagrindiniame registre yra instrukcijos žymė (+) ir du diskretiniai dydžiai, valdymo įrenginys, tiesiogiai arba saugomų tik nuskaitomoje objektinėje atmintyje instrukcijų pagalba, paleidžia skaitmeninį ALU vykdyti instrukciją (sudėtį). Rezulatas išsaugomas kaip kanoninė reikšmė pirmajame pagrindiniame registre (pieš. 8G). Ji yra pažymima apply, kas reiškia, kad uždavinys įvykdytas. Rezultatas - reikšmė 3 - yra siunčiama ir pakeičia visas esamas identifikatoriaus id_1 reikšmes.

Kaip parodyta žemiau, tam tikras registrų steko registrų skaičius saugo sąrašą. Nepanaudoti registrai yra pažymimi kaip laisvi. Sąrašų medis yra naudojamas valdymui ir skaičiavimams atlikti. Šias operacijas kontroliuoja valdymo įrenginys 6 ir, jeigu reikalinga atlikti skaitmenes aritmetines operacijas, kartu dirba ir skaitmenis ALU 5. Skaičiavimai atliekami, perrašant sąrašų medžio turinį.

Sąrašai, turintys iki keturių elementų, saugomi vienoje ląstelės išraiškoje. Galimas darbas ir su ilgais sąrašais, bet tokiu atveju toks sąrašas turi būti sudalintas į mažesnius, kurių ilgis neviršytų galimo apdoroti bazinėje ląstelėje sąrašo ilgio. Bazinė ląstelė vienu metu gali aapdoroti tik tam tikro dydžio medžius. Didesni medžiai bazinėje ląstelėje apdorojami dalimis.

Interfeisas: objektinė atmintis <-> bazinė ląstelė

Objektinė atmintis 1 su bazinė ląstele 2 sujungta transformuojančio interfeiso 7, kuris adaptuoja signalus, ir pakankamai

plačios, kad galėtų vienu metu perduoti visą ląstelės išraiškos turinį, magistralės 8 pagalba. Magistralė 8 susideda iš magistralės dalies OBJv, į kurią įeina linijos id, env, v0, v1, v2, v3, prijungtos prie bazinės ląstelės registrų 3 registrų plokštumų NUM, iš magistralės dalies TAG, prijungtos prie bazinės ląstelės registrų 3 registrų plokštumų HEAD, iš magistralės dalies ATTRIBUTE, įjungiančios linijas iš atinkamų išraiškų laukų plokštumų LAZY, WHERE ir TYPE. Interfeisas 7 sustiprina ir transformuoja iš bazinių registrų 3 magistrale 8 siunčiamus signalus į signalus, suprantamus objektinės atminties ląstelėms. Jis taip pat sustiprina ir transformuoja signalus iš objektinės atminties 1 ląstelių siunčiamus į bazinės ląstelės registrus. Nors parodyta, kad interfeisas randasi objektinėje atmintyje, jis gali būti patalpintas ir bazinėje ląstelėje. Bitinių ląstelių interfeisas parodytas pieš. 20 ir detaliau aprašytas toliau.

ATMINTIES LĄSTELIŲ OPERACIJOS

Pagal pieš. 3, visos atminties ląstelės valdomos iš centrinio valdymo įrenginio 6 siunčiamos valdymo magistrale, prie kurios prijungta bazinė ląstelė 2, skaitmeninės informacijos pagalba.

Kiekvieno išraiškos elemento bitinės ląstelės gali būti valdomos jų viršūnių pagalba, ir atlikti sekančias operacijas:

rest (ilsėtis)	ląstelės saugo savo turimą reikšmę
read (nuskaityti)	saugomos ląstelėje reikšmės yra nuskaitomos
write (įrašyti)	nauja reikšmė įrašoma į ląstelę
compare (sulyginti)	ląstelės reikšmė sulyginama su kita reikšme

Žymė nustatoma prikausomai nuo linijų a ir b būsenų loginės išraiškos, kurios argumentai yra: ankstesnės žymė, sulyginimo operacijos rezultatas ir valdymo įrenginio 6 signalas.

Kadangi naudojamas asociatyvinės atminties adresavimas, t. y. naudojamas saugomas turinys, o ne adresas, neegzistuoja informacijos fizinis pasiekimo būdas. Atmintis yra asociatyvinė. Tokiu būdu paieškai galima naudoti ląstelės identifikatorių, kontekstą, tipą, informacijos reikšmę ar jų kombinacijas.

Atminties ląstelės elemento žymės bitas ar bitai, turinys reikšmę CHOSEN, rodo, kad duotas elementas yra pasiekiamas pasiekimo mechanizmui. Kaip aprašyta žemiau, tam tikros paieškos operacijos

užstato šį bitą.

Pasiekiamos gali būti viena ar kelios atminties ląstelės. Kaip pažymėta aukščiau, viena iš tokių operacijų gali būti įrašymas, kurio metu identifikatorius įrašomas į daugelį elementų.

Linija acc preijungta prie viršūnės 16, sujungia tarpusavyje visas elemento bitines ląsteles. Visos bitinės ląstelės valdomos siganalų, perduodamų linija acc, pagalba. Kitos, prijungtos prie bitinių ląstelių, linijos d ir d^* yra taip prijungtos prie atitinkamų kitų objektinės atminties ląstelių, kas bus paaiškinta vėliau.

Prioritetų dekoderis

Prioritetų dekoderis 11 turi kiekvienai atminties ląstelei skirtas sekcijas. Kiekviena sekcija turi sujungimą REQUEST (REIKALAVIMAS), kur reikšmė 'teisinga' atitinka NEED (REIKALINGA) ir reikšmė 'neteisinga' - NO NEED (NEREIKALINGA), ir sujungimą GRANT (SUTEIKIMAS), kur reikšmė 'teisinga' atitinka CHOSEN (IŠRINKTA) ir reikšmė 'neteisinga' - NOT CHOSEN (NEIŠRINKTA).

Prioritetų dekoderis 11 užstato tik vieną CHOSEN sujungimuose GRANT. Tai gali būti realizuota taip, kad pirmojoje sekcijoje, kuri pareikalus REQUEST lygų NEED, bus užstatyta CHOSEN. Prioritetų dekoderio įranga (pieš. 21) bus aprašyta žemiau.

Kartais reikalinga bendrauti su visomis atminties ląstelėmis. Kelios atminties ląstelės ar išraiškos elementai turi būti nuskaityti ar sutvarkyti. Tai yra daroma prioritetų dekoderio 11 pagalba. Kiekvienai ląstelei, pasiuntusiai REQUEST signalą dekoderis 11 grąžina GRANT signalą.

Atmintis yra valdoma įrašymo, skaitymo ir paieškos operacijomis. Šios operacijos gali būti grupuojamos į sudėtingesnes. Tam tikri paskaičiavimai gali būti realizuojami vidinių a ir b ir bendrų magistralių pagalba.

Paieška realizuojama sulyginimo pagalba, rezultate gaunant FIT (SUTAMPA) arba DIFFERENT (NESUTAMPA). Paieška gali būti realizuota sekančiais būdais:

(1). Paieška yra individuali kiekvienam išraiškos elementui, ir nepriklauso nuo kitų išraiškos elementų

(2). Paieška vykdoma sulyginant visus atminties ląstelės išraiškos elementus. Rezultate kiekviename elemente yra FIT.

(3). Paieška vykdoma sulyginant visus atminties ląstelės išraiškos elementus. Rezultate bent viename išraiškos elemente yra FIT.

Sulyginimas realizuojamas sekančiais būdais:

(1). Du bitiniai šablonai yra lyginami. Sulyginimo rezultatas yra FIT kai visi atitinkami bitai yra vienodi.

(2). Du bitiniai šablonai yra lyginami arba vienas iš jų yra perkoduojamas taip, kad jo šablono informacija atitinka ARBITRARY arba SPECIFIC reikšmę v. Jeigu lyginant viena iš reikšmių atitinka ARBITRARY, rezultatas yra FIT. Kitu atveju FIT bus gaunamas tada, kai dvi informacinės vertės v yra vienodos.

Magistralės funkcija yra objektinės atminties žodžių nuskaitymas ar įrašymas. Magistralė bus yra kontroliuojama pasiekimo access funkcijos pagalba. Pasiekimo funkcija priklauso nuo žymės ir/arba magistralių a ir b reikšmės.

Vienos linijos magistralės a BENDRAS ARBA funkcija susideda iš loginių reikšmių sąrašo. Ji išskaičiuoja loginį ARBA tarp visų ląstelės išraiškos elementų. Fiziškai ji atitinka linijai, kuri tvarkoma tranzistorių grupės, esančios elemento viršūnėje 16, pagalba.

Vienos linijos magistralės a BENDRAS IR funkcija susideda iš loginių reikšmių sąrašo. Ji išskaičiuoja loginį IR tarp visų ląstelės išraiškos elementų. Fiziškai ji atitinka linijai, kuri tvarkoma tranzistorių grupės, esančios elemento viršūnėje 16, pagalba.

Prioritetų dekoderio 11 prioritetinės funkcijos argumentai yra loginių reikšmių sąrašas ir jos rezultatas yra tokio pat dydžio loginių reikšmių sąrašas. Pirmasis argumetų sąrašo elementas turi didžiausią prioritetą. Po šio elemento seka elementai su žemesniais prioritetais. Pirmasis 'argumento teisinga' bitas užstato rezultato atitinkamą 'teisinga' bitą. Visi kiti bitai numetami į 'neteisinga'.

Atminties ląstelių elementai daugiausia naudojami šių elementų

reikšmių paieškai ir toliau atliekant skaitymo ir įrašymo operacijas šiuose elementuose.

BAZINĖS LĄSTELĖS APARATŪRINĖS DALIES STRUKTŪRA

Plokštumų NUM ir HEAD bazinės ląstelės registrai prijungti prie magistralės OBJv linijų, magistralės id, t. y. identifikatoriaus magistralės, magistralės env, t. y. konteksto magistralės, tarp plokštumų 2 ir ir interfeiso 7. Magistralė OBJv įjungia dalis v0, v1, v2, ir v3.

Bazinių registrų ląstelių masyvas tokiu būdu "sudalinamas" statmenai registrams į plokštumas, ir registrų ląstelės esančios toje pačioje NUM ar HEAD plokštumoje, bet skirtinguose baziniuose registruose, tarpusavyje sujungti pagal pieš. 9 parodytą schemą.

NUM ir HEAD plokštumų struktūra, parodyta pieš. 9, yra registrų ląstelių NxN dydžio kvadratinė matrica, susidedanti registrų $S_{0,0} \div S_{N-1,N-1}$.

Baziniai registrai daugelyje atveju naudojami laikinam išraiškų elementų saugojimui. Registrai skirstomi: pagal padėtį- baziniai, pagrindiniai ir papildomi registrai; ir pagal funkciją - sūnaus, uždavinio ir tėvo registrai.

Įrangoje parodytos registrinės matricos dydis yra N=4, bet gali būti pasirinktas ir ktas matricos dydis. Apatinių registrinių ląstelių $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ ir $S_{3,0}$ eilutė, kaip parodyta pieš. 9, prijungta prie magistralės linijos h0 ir pagrindinio registro ląstelių. Pagrindinio registro ląstelės $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ ir $S_{3,0}$ dažnai naudojamos kaip uždavinio šaknies registrai ir yra prijungtos magistralės NU, turinčios tris laidininkus NU1÷NU3, pagalba prie skaitmenio ALU.

Identifikatoriaus registro ląstelė ID prijungta prie linijos id ir konteksto registro ląstelė ENV prijungta prie linijos env.

Magistralės linija hi rakto SW_{vi} pagalba gali būti prijungta prie linijos vi, kur i gali būti nuo 1 iki 3. Magistralė, turinti linijas h0, h1, h2, h3 yra vadinama OBJh, t. y. objekto horizontali magistralė. Magistralė OBJh, be visa ko kito, naudojama duomenims pakrauti vertikaliai, t. y. į bazinės ląstelės, gaunančios duomenis iš objektinės atminties magistralės OBJv pagalba, registrų stulpelius. Tai bus aprašyta toliau pagal pieš. 15.

Magistralės linijos id, env, v0, v1, v2, v3 atitinkamų raktų $SW_{id,h0}$, $SW_{env,h0}$, SW_{v0}, $SW_{v1,h0}$, $SW_{v1,h0}$, $SW_{v1,h0}$ pagalba gali būti prijungtos prie linijos h0. Magistralė res, įjungianti linijas c_{id}, c_f, c_h ir c_v, prijungta prie valdymo įrenginio 6 ir gali būti naudojma konstantės, pvz. 0, užsiuntimui į registrą. Magistralės linija c_{id} prijungta prie identifikatoriaus registro ląstelės, magistralės linija c_f prijungta prie registrų ląstelių F0, F1, F2, ir F3, magistralės linija c_h rakto $SW_{ch,h0}$ pagalba gali būti prijungta prie linijos h0 ir magistralės linija c_v rakto $SW_{vi,cv}$ pagalba gali būti prijungta prie linijos vi, kur i gali būti nuo 1 iki 3.Magistralė res su savo raktais kai kuriais atvejais gali būti ignoruojama.

Papildomi registrai

Duomenų medžio aukščiausias lygis vadinamas tėvu. Tėvas kartais saugomas papildomuose registruose ląstelėse F0, F1, F2, F3. Kiekvienas papildomas registras gali saugoti bazinį žodį. Kiekvienas papildomas registras yra prijungtas prie magistralės linijos id ir vienos iš atitinkamų linijų h0, h1, h2, h3. Papildomi registrai naudojami tik nedaugelyje bazinės ląstelės vykdomų operacijų. Tokiu būdu papildomi registrai tam tikrais atvejais geli būti nenaudojami. reikalui esant, į bazinę ląstelę galima įjungti ir daugiau papildomų registrų.

Kiekvienas registras turi kelias ląsteles veikiančias skirtingose plokštumose. Kaip parodyta pieš. 9 registrai surikiuoti eilutėmis ir stulpeliais. Papildomų registrų sritis F0, F1, F2, F3 yra stulpelyje ir N dydžio registrų sritys $S_{0,0}$-$S_{0,3}$, $S_{1,0}$-$S_{1,3}$, $S_{2,0}$-$S_{2,3}$ ir $S_{3,0}$-$S_{3,3}$ sudaro atitinkamus stulpelius, galinčius saugoti sūnus.

Sujungimai tarp registrų ląstelių

Sujungimai realizuojami tarp atitinkamų registrų ląstelių kiekvienoje plokštumoje tiek vertikaliai, tiek horizontaliai. Sujungimai, turintys kietai užprogramuotas vertes, yra naudojami kiekvienos tolimiausios registrų ląstelės, esančios eilutėje prijungtoje prie objektinės atminties. Ji prijungta prie N sujungimo (North - Šiaurė) registro ląstelėje (pieš. 10) ir naudojams, kai yra daromi postūmiai North-South kryptimi. Diagonaliniai sujungimai tarp registrų ląstelių realizuojami taip, kad

transponojamos pozicijos tampa sujungtomis. Tai reiškia, kad ląstelė $S_{i,j}$, kur i nelygu j, yra sujungta su ląstele $S_{j,i}$. Kiekviena registro ląstelė sujungtas su kaimyne, esančia žemiau ir dešniau. Kiekvienas papildomas, identifikatoriaus, konteksto ir bazinis registras atitinkamai išėjimų ACC_{Fx}, ACC_{id}, ACC_{env} ir $ACC_{Sx,y}$ pagalba yra prijungtas prie vienos iš BOOL plokštumų (x ir y gali būti nuo 0 iki 3).

Žemiau aprašyta bendra registro ląstelė (pieš. 10), kurios organizacijos bendri principai atitinka specifinių registrų, t. y. papildomo, identifikatoriaus, bazinio ir konteksto registrų organizacijos principus. Smulkesnę informaciją apie šiuos specifinius registrus galima rasti US Paraiškoje Nr.....

Bendra registrinė ląstelė

Pagal pieš. 10, registro ląstelė įjungia dvi vidines magistrales a_R ir b_R ir centrinį vidinį registrą r_R. Vidinės magistralės a_R ir b_R turi ryšį ir su išorinėmis grandinėmis. Kaip parodyta pieš. 10, bendra registro ląstelė čia turi visus galimus išorinius ryšius. Paprastai specifinio registro ląstelė nepalaiko visų ryšių, kaip pieš. 10, priklausomai nuo ląstelės padėties kai kurie iš jų neegzistuoja. Sujungimai tarp terminalų yra parodyti pieš. 9. Kaip parodyta pieš. 9, ne visos registro ląstelės turi išorinius ryšius, parodytus pieš. 10. Smulkesnė informacija apie registrų ląsteles čia nebus pateikta.

Magistralė a_R

Magistralė a_R prijungta prie vertikalios magistralės linijos vx, kur x yra nuo 0 iki 3, rakto SW_{Vi} ir terminalo V pagalba. Magistralė a_R taip pat prijungta prie horizontalios magistralės linijos hy, kur x yra nuo 0 iki 3, rakto SW_{Hi} ir terminalo H pagalba. Ji taip pat prijungta per terminalą W (West - Vakarai) prie kaimyninės ląstelės SW_E rakto (East - Rytai), ir, jei registro ląstelė yra pagrindinio registro ląstelė, per terminalą NU prijungta prie skaitmenio aritmetinio įrenginio lp. Magistralė a_R taip pat per terminalą Da prijungta prie žemiau ir dešnėje esančios ląstelės rakto SW_{Db} ir, per terminalą S (South - Pietūs), prie apatinės ląstelės rakto SW_N (North -Šiaurė). Registro ląstelė per terminalą C ir raktą SW_C, prijungtus prie magistralės a_R, gali būti numesta arba užstatyta. Taip pat magistralė a_R per raktą SW_{a1}

prijungta prie centrinio vidinio registro r_R įėjimo, o per raktą SW_{a0} - prie jo išėjimo.

Magistralė b_R

Magistralė b_R per raktą SW_E ir terminalą E yra prijungta prie ląstelės dešinėje, per raktą SW_{Db} ir terminalą Db - prie diagonalinio kaimyno ir per raktą SW_N ir terminalą N - prie viršutinės registro ląstelės. Magistralė b_R taip pat per raktą SW_{b1} prijungta prie centrinio vidinio registro r_R įėjimo, o per raktą SW_{b0} - prie jo išėjimo.

Centrinis vidinis registras

Centrinis vidinis registras r_R įjungia du CMOS tipo invertorius Q1 ir Q2, ir valdomą raktą SW_Q, esantį tarp invertorių. Pilna registro ląstelė taip pat įjungia magistrales a_R ir b_R bei raktus SW_{a1}, SW_{a0}, SW_{b1}, SW_{b0} ir raktus, jungiančius ląstelę su išore. Centrinio vidinio registro r_R išėjimas gali būti prijungiamas prie horizontalių ir vertikalių magistralių atitinkamai rakto SW_{Ho} ir terminalo H bei rakto SW_{Vo} ir terminalo V pagalba. Centrinis vidinis registras r_R saugo dinaminę būseną (paaiškinta žemiau).

Raktinės operacijos

Visi valdomi raktai visuose registrų ląstelėse valdomi valdyno įrenginio 6, turinčio daug loginių elementų, pagalba. Šis loginis masyvas saugomą bazinėje ląstelėje informaciją naudoja raktų valdymui. Loginių elementų darbas sinchronizuojamas taktiniu dažniu. Raktai yra dvikrypčiai, bet kai kurie veikia tik viena kryptimi. pvz. iėjimo ir išėjimo raktai SW_{Hi} ir SW_{Ho}.

Komparatoriaus įrenginys COMP

Komparatoriaus įrenginys COMP įjungia pirmąjį NE-IR loginį elementą G1. Vienas įėjimas yra prijungtas prie invertoriaus Q1 neinvertuojančio įėjimo o kitas prie invertoriaus Q2 įėjimo. Įrenginys taip pat turi kitą NE-IR loginį elementą G2. Vienas įėjimas yra prijungtas prie invertoriaus Q1 išėjimo, o kitas prie invertoriaus Q2 išėjimo. Loginių elementų G1 ir G2 išėjimai yra prijungti prie vieno laido magistralės ACC vedančios į vieną iš BOOL plokštumų. Abu NE-IR loginiai elementai realizuoti dvejomis MOP lauko tranzistorių, kurių ištakai/santakai grandinės prijungtos tarp žemės ir maitinimo įtampos ir kurių užtūros yra NE-IR loginio

elemento įėjimai o viršutinių MOP lauko tranzistorių santakai yra išėjimai, grupėmis. Šis komparatoriaus įrenginys COMP yra naudojamas asociatyvinei paieškai atlikti, t. y. kai elementas esantis bazinėje ląstelėje turi būti palygintas su elementu objektinėje atmintyje arba su elementu kitoje bazinės ląstelės dalyje. Tada elementas kuris turi būti sulygintas prijungiamas prie registro ląstelių, kuriose yra etalonas, išėjimų (bus aprašyta toliau).

Invertoriai ir raktai

Invertoriai Q1 ir Q2 gali būti realizuoti arba dvejais MOP lauko tranzistoriais, pvz. dvejais komplementariniais MOP lauko tranzistoriais. Valdomo registro ląstelės raktai gali būti realizuoti arba MOP lauko tranzistoriumi arba dviem komplementariniais MOP lauko tranzistoriais. Valdymo įrenginys 6 raktus valdo valdymo signalų pagalba. Raktas, tam kad pagreitinti būsenų pasikeitimą, gali būti valdomas tiek valdymo signalais, tiek savo papildomais signalais.

Asociatyvinė paieška ir BOOL plokštuma (plokštumos)

Asociatyvinės paieškos metu pasiekimo magistralės, einančios į BOOL plokštumą, pagalba atliekama sulyginimo operacija. Du IR loginiai elementai G1 ir G2 sulygina raktų reikšmes, t. y. reikšmę, kurią reikia sulyginti (ji yra Q1 įėjimuose), ir reikšmę, esančią Q2 įėjimuose. Šios lyginimo operacijos metu rakto reikšmė perduodama magistralių a_R ir b_R pagalba į Q1. Raktas SW_Q turi būti atidarytas (neįjungtas). Jeigu lyginamos reikšmės nesutampa, pakrauta BOOL plokštuma iškraunama per NE-IR loginius elementus G1 ir G2. Jeigu sutampa, BOOL plokštuma lieka užkrauta.

Visos magistralės linijos ACC registre (viena linija - vienai registro ląstelei), gali būti lygiagrečiai sujungtos ir prijungtos prie tos pačios magistralės linijos BOOL plokštumoje. Kaip alternatyva, visų registrų ląstelių ACC linijos vedančios į NUM ir HEAD plokštumas gali būti prijungtos prie magistralės linijos, skirtos šioms plokštumoms, esančios BOOL plokštumoje, ir visos registrų ląstelės esančios ATTRIBUTE plokštumoje ali būti prijungtos arba prie tos pačios BOOL plokštumos arba prie sekančios BOOL plokštumos, skirtos ATTRIBUTE plokštumai. Esant daugiau kaip du BOOL plokštumos ir egzistuoja viena ar dvi linijos, pagal

valdymo instrukcijas, saugomos valdymo įreginyje 6, svarbu jas išsirinkti. Ši galimybė numatyta išradime. BOLL plokštumų skaičius apsprendžia asociatyvinės paieškos tikslumą, t. y. galimas skirtingas asociatyvinės paieškos būdų skaičius. Tokiu būdu sulyginimas atliekamas vienu ypu registro daliai, prijungtai prie tokios pačios BOOL plokštumos magistralės linijos. Jei visi NE-IR loginiai elementai išėjime turi tą patį (aukštą) lygį, tada sulyginimo rezultatas yra "sutampa"; kitu atveju rezulattas yra "nesutampa", kur "sutampa" reiškia, kad abu informacijos vienetai yra identiška. Plokštumos BOOL tokiu būdu yra magistralių linijų plokštumos ir gali būti laikomos kaip virtualios plokštumos.

ATTRIBUTE plokštumų konfigūracija

ATTRIBUTE plokštumos gali turėti skirtingą konfigūraciją, negu plokštumos NUM ir HEAD, jos pvz. gali būti realizuotos kaip registrų eilutė, o ne matrica. ATTRIBUTE plokštumos gali turėti papildomą magistralės liniją, kuri gali būti prijungta prie visų ar kelių plokštumos registrų ląstelių, ir kuri prijungta prie valdymo įrenginio 6, kuris gali naudoti šios magistralės infornaciją redukavimo operacijos tipui nustatyti. ATTRIBUTE plokštumos nereikaluja konteksto registrų ląstelių palaikymo. Magistralės linijos v0, v1, v2 ir v3 prijungtos prie kitų interfeiso 7 įėjimų, negu analogiškos linijos NUM ir HEAD plokštumose ir atitinkamai prie kitų objektinės atminties 1 dalių, tokių kaip lazy (aktyvumas), where (kur) ir type (tipas) (pieš. 1). Kaip alternatyva, magistralės linijos v0, v1, v2 ir v3 gali būti neprijungtos prie objektinės atminties 1. To vietoje magistralės linija id ATTRIBUTE plokštumoje gali būti panaudota objektinės atminties būsenos, t. y. lazy, where ir type, perdavimui į atitinkamą plokštumą bazinėje ląstelėje.

Atminties įrašymo režimas

Atminties saugojimo režimas yra suformuojamas vienu arba dvejais registro ląstelėje vykdomais ciklais. Vienas ciklas suformuojamas rakto SW_{b0}, magistralės b_R, rakto SW_{b1}, invertoriaus Q1, rakto SW_Q ir invertoriaus Q2 pagalba. Kitas ciklas suformuojamas rakto SW_{a0}, magistralės a_R, rakto SW_{a1}, invertoriaus Q1, rakto SW_Q ir invertoriaus Q2 pagalba. Kai raktai viename ar kitame cikle uždaryti, signalas perduodamas per invertorius Q1 ir Q2 ir signalo

lygis tampa patovus invertoriaus Q1 įėjime ir invertoriaus Q2 išėjime - taip duomenys įrašomi į registro ląstelę. Ląstelė saugo dinaminę būseną.

Išėjimo režimas

Išėjimo režimo metu, Q2 išėjimas gali būti perduotas į vieną iš magistralių a_R arba b_R, ir raktų pagalba gali būti perduotas į vieną ar daugiau išėjimo terminalų (N, S, E, W, ir kt.). Kita magistralė a_R arba b_R gali būti naudojama pasirenkmam režime. Jei raktas SW_Q yra atidarytas, invertoriaus Q2 išėjimas yra stabilus, t. y. nebus pakeistas, kol raktas SW_Q bus atidarytas. Invertoriaus išėjimas gali būti perduotas per uždarytą raktą SW_{b0} į magistralę b_R ir per uždarytą raktą SW_{a0} į magistralę a_R. Infornacija iš magistralių a_R ir b_R gali būti perduota į išorines magistrales, prie kurių prijungta registro ląstelė, valdant raktus, esančius tarp ląstelių.

Įėjimo režimas

Įėjimo režime vienas iš raktų SW_{a1} arba SW_{b1} yra uždarytas. Tokiu būdu vieno iš terminalų (N, S, E, W, ir kt.) būsena yra perduodama į vidines magistrales a_R arba b_R ir iš jų į centrinį vidinį registrą r_R.

Perdavimai

Yra įmanoma dviejų fazių ciklo eigoje perduoti terminalų pagalba duomenis iš bazinės ląstelės vienos registro ląstelės į kitą. Dviejų registrų ląstelių duomenų apsikeitimas vertikalia, horizontalia arba diagonaline kryptimi galimas per trijų fazių ciklą.

Raktas SW_Q yra taktuojamas tiesiai iš taktinio generatoriaus, ir tai daroma vienu metu visose registrų ląstelėse, todėl perdavimas tarp invertorių Q1 ir Q2 atliekamas vienu metu visoje bazinėje ląstelėje. Kiti raktai yra valdomi skirtingomis taktinio dažnio fazėmis. Taktinis generatorius naudojamas visoms bazinės ląstelės operacijoms sinchronizuoti.

Taktinis ciklas dalinamas į takto fazę 0, a ir/arba b. Fazė 0 yra pirmoji fazė, t. y. fazė kai centrinis vidinis registras r_R yra saugojimo režime - kai duomenys yra nesikeičia. Fazė a naudojama perduodant duomenis iš magistralės a_R, o fazė b naudojama

perduodant duomenis iš magistralės b_R.

Vienos krypties perdavimas, t. y. tik į arba iš registro ląstelės, užima dviejų fazių ciklą. Pirmoji fazė 0 yra stabili. Sekanti fazė a arba b naudojama perdavimui.

Dviejų krypčių perdavimo atveju, t. y. perdavimas vyksta sukeičiant ląstelių turinius, duomenų perdavimas trunka trijų fazių ciklą. Pirmoji fazė 0 yra stabili. Fazių a ir b metu perdavimas vyksta skirtingomis kryptimis.

Reikia pažymėti, kad išradimas liečia ir daugiau fazių turinčius perdavimo ciklus, pvz. turinčius dvi b fazes.

Raktai SW_{a1} ir SW_{b1} normaliai yra uždaryti. Abi lokalinės magistralės a_R ir b_R saugo registro ląstelės saugomas reikšmes. Kai į lokalinės magistralės a_R arba b_R įėjimą reikia paduoti naują reikšmę saugojimui, atitikamas raktas SWa1 arba SWb1 yra atidaromas. Trumpam uždaromas raktas, jungiantis su išorinėmis (horizontaliomis ar vertikaliomis) magistralėmis, ir informacija patenka į vidinę magistralę.

Taip pat galima panaudoti postūmio tinklą, t. y. tinklą tarp skirtingų registrų ląstelių, įjungiantį prie terminalų prijungtus raktus, registro ląstelės turinio pernešimui kuria nors kryptimi (N, arba S, arba W, arba E).

Vienos krypties perdavimo operacijos pavyzdys

Pieš. 11A parodyta dvi kaimyninės bazinės ląstelės. Duomenys iš kairiosios ląstelės, siuntėjo, perduodami į dešniąją, gavėją. Įrenginio 6 kontroliniai signalai perjunginėja raktus. Pieš. 11B rodo kiekvieno rakto padėtį kiekvienos perdavimo fazės metu, kur mažesnę reikšmė rodo atidaryto rakto padėtį, o didesnė - uždaryto rakto padėtį. Pats perdavimas vyksta fazės b metu. Perdavimas realizuotas sekančiu būdu (skirtingi etapai žymimi tais pačiais numeriais pieš. 11A ir 11B):

0. Grandinė yra stabilioje būsenoje, SW_Q, SW_{a0}, SW_{a1}, SW_{b0}, SW_{b1} uždaryti, visi kiti raktai tiek siuntėjuje, tiek gavėjuje yra atidaryti (šis etapas pieš. 11A nepažymėtas, kadangi tai liečia visus raktus). Ši stabili būsena atitinka fazę 0 pieš. 11B,

1. Pirmos fazės (fazė b), trunkančios vieną periodą, metu kai

raktas SW_Q tiek siuntėjuje, tiek gavėjuje atidaromas,

2. raktas SW_{a0} atidaromas ir raktas SW_{b0} uždaromas tiek siuntėjuje, tiek gavėjuje
3. raktas SW_E tarp siuntėjo ir gavėjo uždaromas
4. raktas SW_{b1} atidaromas tiek siuntėjuje, tiek gavėjuje, ir
5. raktas SW_{a1} siuntėjuje atidaromas ir raktas SW_{a1} gavėjuje uždaromas. Tai leidžia pernešti duomenis iš siunėjo vidinio registro į gavėjo vidinį registrą.
6. Antros fazės (fazė 0) takto metu, kai raktas SW_{b1} uždaromas tiek siuntėjuje, tiek gavėjuje,
7. raktas SW_E tarp siuntėjo ir gavėjo atidaromas,
8. raktai SW_{b0} ir SW_{a0} uždaromi pirmiausia ir tokiu būdu raktai SW_{b1} ir SW_{a1} tiek siuntėjuje, tiek gavėjuje taip pat uždaromi. Tokiu būdu grandinė gražinama į stabilią būseną, aprašytą etape 0, t. y. fazę 0.

Dviejų krypčių perdavimo operacijos pavyzdys

Pieš. 12A parodytos dvi registro ląstelės ir duomenys, kurie tarp dviejų skirtingų registro ląstelių pernešami dviejų krypčių pernešimo operacijos metu. Valdymo įrenginio 6 signalai valdo raktų būseną. Pieš. 12B parodyta kiekvieno rakto būsena skirtingose perdavimo fazėse, kur žemesnioji reikšmė atitinka atidarytą rekatą, o aukštesnioji - uždarytą. Abi registrų ląsteleles durba ir kaip siuntėjas ir kaip gavėjas, tolėl toliau jos bus vadinamos "ląstele 1" ir "ląstele 2". Vienas perdavimas, iš ląstelės 2 į ląstelę 1 vyksta fazės a metu ir kitas perdavimas, iš ląstelės 1 į ląstelę 2 vyksta fazės b metu. Skirtingi perdavimo etapai žymimi tais pačiais numeriais pieš. 12A ir 12B. Perdavimas realizuojamas sekančiu būdu:

0. Grandinė yra stabilioje būsenoje, SW_Q, SW_{a0}, SW_{a1}, SW_{b0}, SW_{b1} uždaryti, visi kiti raktai abiejose ląstelėse yra atidaryti (šis etapas pieš. 11A nepažymėtas, kadangi tai liečia visus raktus). Ši stabili būsena atitinka fazę 0 pieš. 12B,
1. Pirmos fazės metu (fazė a), kai raktas SW_Q ląstelėse 1 ir 2 yra atidarytas,
2. raktas SW_{a0} ląstelėse 1 ir 2 yra uždarytas ir raktas SW_{b0}

ląstelėse 1 ir 2 yra atidarytas,

3. raktas SW_E tarp ląstelių uždarytas,

4. raktas SW_{a1} ląstelėse 1 ir 2 atidarytas, ir

5. raktas SW_{b1} ląstelėje 1 yra uždarytas ir raktas SW_{b1} ląstelėje 2 yra atidarytas. Tai leidžia perduoti duomenis iš ląstelės 2 į ląstelę 1.

Sekančios fazės metu (fazė b), kai raktas SW_Q vis dar atidarytas,

6. raktas SW_{a0} ląstelėse 1 ir 2 yra atidarytas ir raktas SW_{b0} ląstelėse 1 ir 2 yra uždarytas,

7. raktas SW_{b1} ląstelėse 1 ir 2 yra atidarytas, ir

8. raktas SW_{a1} ląstelėje 1 yra atidarytas ir raktas SW_{a1} ląstelėje 2 yra uždarytas, ko pasekoje galimas duomenų perdavimas iš ląstelės 1 į ląstelę 2.

9. Trečios fazės (fazė 0) metu, kai raktas SW_Q abiejose ląstelėse 1 ir 2 yra uždarytas,

10. raktas SW_E ląstelėje 1 atidaromas, ir

11. pirma uždaromi raktai SW_{b0} ir SW_{a0}, po to uždaromi raktai SW_{b1} ir SW_{a1} abiejose ląstelėse. Tokiu būdu grįžtama į pradinę stabilią padėtį, aprašytą etape 0, t. y. fazę 0.

Raktų SW_{a0} ir SW_{b0} valdymo signalai.

Signalai yra įjungti fazės 0 metu. Visose lokalinėse magistralėse laikoma saugoma reikšmė. Įėjimui naudojama magistralė valdoma atidarant raktus SW_Q ir SW_{x0}, kur x yra a arba b. Įvedimo metu keletas magistralių gali būti užtrumpintos terminalais (E, V, D, H ir kt). Tuojau po to magistralėse atstatoma teisinga reikšmė.

Susidaro tam tikras valdymo signalo krintančio fronto užlaikymas tarp raktų SW_Q ir SW_{x0} (x yra a arba b). Jeigu jis trumpas, nesusidaro problemų. Bet jeigu jis pasieka milisekundines reikšmes, magistralė x_R (x yra a arba b) praranda savo dinamiškai saugomą reikšmę.

Susidaro tam tikras valdymo signalo kylančio fronto užlaikymas tarp raktų SW_Q ir SW_{x0} (x yra a arba b). Jeigu jis yra neigiamas magistralėje x_R tarp invertorių Q1 ir Q2 atsiras neteisinga vertė. Todėl naudojamas teigiamas užlaikymas.

Raktų SW_E, SW_V, SW_D ir SW_H ir kt. valdymo signalai.

Raktai normalioje būsenoje yra atidaryti. Tokiu būdu visos lokalinės magistralės yra atjungos nuo išorės. Išėjimui arba įėjimui naudojama magistralė yra valdoma signalų pagalba uždarant terminalinius raktus. Įvedimo arba išvedimo metu keletas magistralių gali būti trumpam užtrumpintos kai kuriais raktais (SW_A, SW_V, SW_D, SW_H ir kt). Tuojau po to magistralėse atstatoma teisinga reikšmė.

Susidaro tam tikras užlaikymas tarp valdymo signalo krintančio fronto rakte SW_Q ir valdymo signalo kylančio fronto rakte SW_Z (Z gali būti H, D, N, V, E ir kt., t. y. bet kuris terminalas per raktą prijungtas prie vidinės magistralės a_R arba b_R). Jeigu jis yra neigiamas, lokalinės magistralės x_R (x yra a arba b) reikšmė gali pasikeisti. Todėl šis laikinis užlaikymas turi būti teigiamas.

Susidaro tam tikras užlaikymas tarp valdymo signalo kylančio fronto rakte SW_Z (Z gali būti H, D, N, V, E ir kt., t. y. bet kuris terminalas per raktą prijungtas prie vidinės magistralės a_R arba b_R) ir valdymo signalo krintančio fronto rakte SW_{X1}. Jeigu jis yra neigiamas, reikšmė negali patekti į įėjimą. Todėl naudojamas teigiamas laikinis užlaikymas.

Susidaro tam tikras užlaikymas tarp valdymo signalo kylančio fronto rakte SW_{X1} ir valdymo signalo krintančio fronto rakte SW_Z. Jeigu jis yra neigiamas, lokalinės magistralės būsena gali pasikeisti, ir registre gali atsirasti klaidinga reikšmė. Todėl naudojamas teigiamas laikinis užlaikymas.

Raktų SW_{a1} ir SW_{b1} valdymo signalai.

Signalai yra įjungti fazės 0 metu. Gali susidaryti nedidelis laikinis valdymo signalų kylančio fronto užlaikymas tarp raktų SW_Q ir SW_{X1} (x yra a arba b). Jeigu jis yra neigiamas, invertoriaus Q2 įėjimas negali dirbti su magistrale x_R (x yra a arba b). Todėl naudojamas teigiamas užlaikymas.

Skaičiavimo mechanizmas bazinės ląstelės

Tam tikras sąrašas instrukcijų atliekamas vieno mašininio ciklo metu.

Kaip pažymėta aukščiau, bazinė ląstelė vykdo struktūrinę

aritmetiką. Visi vykdymo žingsniai atliekami bazinėje ląstelėje naudojant sąrašines instrukcijas. Insrukcijų pvyzdžiai duoti žemiau:

length	(ilgis) apskaičiuojamas uždavinio ilgis
map	(planas) funkcija dirba su sąrašo elementais. Jeigu sąraše yra įstatytų sąrašų, ši instrukcija dirba ir su šių įstatytų sąrašų elementais. (Ši instrukcija bus paaiškinta žemiau)
filter	(filtruoti) funkcija filtruoja sąrašo elementus. Funkcija taip pat taikoma įstatytiems sąrašams.
join	(sujungti) visi elementai perrašomi į įstatyto sąrašo elementus. Funkcija taip pat dirba ir su įstatytais sąrašais.
transpose	(transponuoti) transponuojama nedidelė matrica. jeigu joje yra sąrašiniai elementai, jie yra pakeičiami. Tvarkomi ir įstatyti sąrašai. (Transponavimo instrukcija bus paaiškinta toliau)

ir t. t.

Bazinės ląstelės atmintis

Bazinėje ląstelėje saugoma

- uždavinys, kuris turi būti redukuojamas, saugomas keliuose registruose, dažniausiai baziniuose registruose
- tam tikrais atvejais, pvz. kai redukuojama trijų lygių struktūra, uždavinio šaknis saugoma dažniausiai papildomuose registruose ir likusi struktūros dalis saugoma bazinių registrų matricoje

Objektinė atmintis 1 gali saugoti tik vieno lygio struktūrą, tuo tarpu kai bazinės ląstelės registrai 3 gali saugoti trijų lygių struktūrą. Yra keturi laikino saugojimo bazinėje ląstelėje atvejai, t. y. kai saugomas 0, 1, 2 arba 3 lygių objektas. Jeigu saugomas trijų lygių objektas, tiktai šaknis ir viena jos atšaka yra saugomi. Visais kitais atvejais visi lygiai yra sugomi.

Paprastas medis, t. y. medis su vienetine reikšme (0 lygio objektas), yra saugomas pirmajame pagrindiniame registre.

Vieno lygio medis yra saugomas pagrindiniuose registruose.

Dviejų lygių medžio šaknies sąrašas, t. y. tėvas, gali būti saugomas horizontaliai pagrindiniuose registruose, ir sąrašai, kurie yra sūnūs, saugomi vertikaliai baziniuose registruose. Kaip alternatyva, šaknis gali būti saugoma papildomuose registruose, o vienas iš jos sūnų - pagrindiniuose registruose. Reikia pažymėti, kad saugojimo būdą, priklausomai operacijos tipo, parenka valdymo įrenginys 6.

Trijų lygių medžio šakninis sąrašas saugomas viename iš papildomų registrų, o vienas iš jo dviejų lygių sūnų saugomas bazinių registrų matricoje.

Tokiu būdu, uždavinio medžio šaknies sąrašas, priklausomai nuo medžio lygių skaičiaus ir vykdomos operacijos tipo, gali būti saugomas skirtinguose bazinės ląstelės registrų vietose.

Uždavinio medžio šaknis yra redukuojama, kaip unify, išraiška. Vykdomoje funkcijoje (apply...) pirmasis elementas yra arba instrukcija, arba identifikatorius, nurodantis į išraiškos struktūrą, naudojamą kaip funkcijos apibrėžimą; visi kiti elementai yra instrukcijos/funkcijos abibrėžimo argumentai.

BAZINĖS LĄSTELĖS ATMINTIES VALDYMAS

Saugoma bazinės ląstelės registruose informacija pakraunama iš objektinės atminties 1. Informacija bazinės ląstelės registruose saugoma sekančiais būdais:

Bazinės ląstelės HEAD ir NUM plokštumų registrai prijungti prie objektinės atminties magistralės OBJ, pasiekimo magistralės ACC, magistralės res ir skaitmeninio ALU magistralės NU. Saugoma būsena susideda iš: dviejų ID ir ENV registrų, papildomų registrų F0÷F3 ir bazinių registrų $S_{0,0}$÷$S_{3,3}$ saugomų būsenų. Bazinės ląstelės ATTRIBUTE plokštumos registrai prijungti prie objektinės atminties panašiai, tačiau jie jungiasi prie kitų objektinės atminties 1 dalių (prie lazy, where ir type dalių).

Bazinės ląstelės valdymo žodis susideda iš valdymo žodžių, skirtų raktams SW_{vi}, $SW_{vi,cv}$, kur i yra nuo 0 iki 3, $SW_{id,h0}$, $SW_{ch,h0}$, $SW_{env,h0}$, $SW_{v1,h0}$, $SW_{v2,h0}$, $SW_{v3,h0}$, SW_{vi0}, SW_{v1}, SW_{v2} ir SW_{v3}, registrams ID ir ENV, papildomiems registrams F0÷F3 ir baziniams registrams $S_{0,0}$÷$S_{3,3}$.

Valdymo žodžiai yra perduodami iš valdymo įrenginio 6 kelių valdymo laidų pagalba. Valdymo laidai gali būti dvifazės porinės valdymo linijos arba parastos vieno laido vienos fazės linijos - tai priklauso nuo raktų, į kurias jos ateina, tipo.

Kievienai bazinės ląstelės registro ląstelei skirtas valdymo žodis susideda iš bendrosios dalies ir specifinės dalies, skirtos būtent tam registrui valdyti. Bendrosios dalies pagalba valdomi bazinės ląstelės raktai SW_{a0}, SW_{b0} ir SW_Q (pieš. 10). Reikia pažymėti, kad be šioje aprašytoje įrangoje scheminių sprendimų galimi ir alternatyvūs variantai.

BAZINĖS LĄSTELĖS OPERACIJŲ PAVYZDŽIAI

Pieš. 13 - 19 yra pieš. 9 išplėtimai. Pieš. 9 nuorodos tinka ir pieš. 13÷19. Pieš. 13 - 19 aprašymuose registrų ląstelių paaiškinimai tinka visam bazinės ląstelės tos plokštumos registrui.

1. Objektinės atminties pasiekimas

Instrukcija mpx_mv:

Objektinės atminties operacijos mpx_mv metu objektinė atmintis 1 yra nuskaitoma ir užstatomos kai kurių bazinių registrų reikšmės. Pasiekiamas objektas yra perduodamas magistralėmis v0, v1, v2, v3 į pagrindinius registrus $S_{0,0}$, $S_{1,0}$, $S_{2,0}$, $S_{3,0}$, magistrale id į registrą ID ir magistrale env į registrą ENV (pieš. 13 tai parodyta storomis linijomis su rodyklėmis). Tuo pat metu pagrindinių registrų senasis turinys yra išsaugomas objektinėje atmintyje 1 kaip išraiška. Tokiu būdu instrukcija mpx_mv išsaugo esamą bazinės ląstelės išraišką objektinėje atmintyje ir pakrauna sekančią vykdomą objektinės atminties išraišką į bazinę ląstelę.

Instrukcija fetch:

Pieš. 14 ir 15 parodyta situacija, kai identifikatorių, saugomą viename iš pagrindinių registrų reikia pakeisti informacija, kurią jis aprašo. Identifikatoriaus, pvz. saugomo $S_{2,0}$, reikšmė perduodama į objektinę atmintį 1, objektinė atmintis suranda patį identifikatorių ir turinį, kurį jis aprašo, turinys magistralės linijų v0÷v3 pagalba pkraunamas į vertikalų stulpelį į registrus, pvz. $S_{2,0}$÷$S_{2,3}$ (pieš. 15).

Operacija prasideda perduodant registre $S_{2,0}$ esantį identifi-

katorių per magistralę h0 ir raktą $SW_{id,h0}$ į vertikalią magistralę į registrą $S_{2,0}$ (pieš. 14). Saugoma reikšmė gali tokiu pat būdu būti perduota ir iš kitų registrų.

Operacija toliau tęsiama (pieš. 15), pakraunant objektinės atminties 1 reikšmes, esančias magistralės linijose v0, v1, v2, v3, į atinkamus registrus, kurie pvz. gali būti $S_{2,0}$, $S_{2,1}$, $S_{2,2}$, $S_{2,3}$, per raktus SW_{v0}, SW_{v1}, SW_{v2}, SW_{v3}, ir magistrales h0, h1, h2, h3.

Objektinės atminties operacijos make ir unify_id gali būti naudojamos bazinės ląstelės turiniui įrašyti į objektinę atmintį.

Instrukcija make:

Pirmojo operacijos make etapo metu reikalingų registrų turinys yra perduodamas kaip pieš. 15, bet atvirkščia kryptimi. Operacijos metu taip pat pernešamas konteksto registro turinys. Atliekama asociatyvinė paieška pernešamui objekui objektinėje atmintyje surasti. Jeigu objektas surandamas, grąžinamas jo identifikatorius, kitu atveju gražinamas laisvas (nepanaudotas) identifikatorius. Abiejais atvejais grąžinamas identifikatorius perduodamas į bazinę ląstelę magistralės linijos id pagalba. Kaip alternatyva identifikatorius gali būti grąžinamas į naudojamo registrų stulpelio pagrindinį registrą. Tokiu sukuriamas asociatyvinis ryšys tarp identifikatoriaus ir bazinės ląstelės turinio.

Instrukcija unify_id:

Operacija unify_id parodyta pieš. 16 ir jos metu viename iš registrų, pvz. $S_{2,0}$, esantis identifikatorius perduodamas į visas vertikalias magistrales id, env, v0, v1, v2, v3 sujungiant reikiamą registro ląstelę su horizontalia magistrale h0 ir prijungiant visas vertikalias magistrales su magistrale h0 raktų $SW_{id,h0}$, $SW_{env,h0}$, SW_{v0}, SW_{v1} ir kt. pagalba. Ši poeracija gali būti naudojma asociatyvinei paieškai ir pakeitimui, pvz. surandant visus tam tikro identifikatoriaus pasikartojimus ir pakeičiant šį identifikatorių nauja, redukuota reikšme.

Panaši į operaciją unify_id pirmame žingsnyje gali naudoti instrukciją make bazinės ląstelės turinio identifikatoriaus gavimui ir anrajame žingsnyje šį turinį pasiųsti į magistralės linijas prijungtas prie objektinės atminties, tam kad objektinėje atmintyje būtų išsaugotas identifikatorius ir jį atitinkantis turinys.

Pavyzdys yra parodytas priede 1. Ten parodytas bazinės ląstelės turinys ir raktų padėtys fazėse a, b ir 0.

2. Skaitmeniniai redukavimai

Skaitmenino redukavimo metu redukuojamas objektas, t. y. uždavinys yra patalpinamas į pagrindinius registrus. Bendru atveju visas uždavinys dalyvauja redukavimo procese. Paprastai pagrindiniame registre $S_{0,0}$ saugomas instrukcijos kodas, kurio bitinis piešinys (paternas) yra skirtingas skirtingoms instrukcijoms. Registrai $S_{1,0}$ ir $S_{2,0}$ naudojami diadinėse operacijose, t. y. opracijose su dvejais operandais, o registras $S_{1,0}$ naudojamas monadinėse operacijose, t. y. operacijose su vienu operandu. Sekantys registrai naudojami sąrašams, jų turinys po redukavimo yra išstumiamas į kairę.

Pagrindinės skaitmeninės operacijos vyksta naudojant registrus $S_{1,0}$ ir $S_{2,0}$. Pagrindinis skaitmeninio ALU sumatorius yra prijungtas prie šių dviejų registrų. Kiti registrai gali būti panaudojami tokiose instrukcijose, kaip mul, div ir mod.

Gali būti naudojami sekantys instrukcijų tipai:

monadinės intrukcijos

registre $S_{0,0}$ yra instrukcija, o registre $S_{1,0}$ yra operandas. Registrai $S_{2,0}$ ir $S_{3,0}$ nenaudojami. Skaitmeninio ALU rezultatas grąžinamas į visus pagrindinius registrus. Nekonvejeriniame režime jis išsaugomas registre $S_{1,0}$. Konvejeriniame režime jis išsaugomas arba papildomame registre arba baziniame registre.

diadinės instrukcijos

registre $S_{0,0}$ yra instrukcija, o registruose $S_{1,0}$ ir $S_{2,0}$ yra operandai. Registras $S_{3,0}$ nenaudojamas. Skaitmeninio ALU rezultatas grąžinamas į visus pagrindinius registrus. Nekonvejeriniame režime jis išsaugomas registre $S_{1,0}$. Konvejeriniame režime jis išsaugomas arba papildomame registre arba baziniame registre.

mul, div, mod instrukcijos

registre $S_{0,0}$ yra instrukcija, o registruose $S_{1,0}$ ir $S_{2,0}$ yra operandai. Galutinis rezultatas išsaugomas registre $S_{1,0}$.

unify redukavimas

Šios operacijos metu skaitmeninio ALU pagalba registro $S_{0,0}$ turinys sulyginamas su registro $S_{1,0}$ turiniu. Operacijos metu gali būti naudijami ir kiti pagrindiniai registrai. Požymių žodžiai, išsaugoti registrų HEAD plokštumoje, kartu su sulyginimo opercijos rezultatais naudojami tolimesniuose skaičiavimuose.

Instrukcijos mul, div ir mod vyksta tiktai skaitmeniniame ALU. Tarpinės reikšmės gali būti laikinai saugomos linijose, jungiančiose skaitmeninį ALU su bazinės ląstelės pagrindiniais registrais, t. y. magistralėje NU.

3. Struktūros redukavimai (minimizavimai)

Struktūros redukavimo metu redukuojamas objektas, t. y. uždavinys patalpinamas į pagrindinius registrus. Bendru atveju redukavime dalyvauja visi ar keli pagrindiniai registrai. Paprastai pagrindiniame registre $S_{0,0}$ saugomas instrukcijos kodas, kurio bitinis piešinys (paternas) yra skirtingas skirtingoms instrukcijoms.

Instrukcija map susideda iš funkcijos f ir sąrašo (e_1, ..., e_n), kuris yra argumentas, ir pritaiko funkciją kiekvienam sąrašo elementui. Rezultate grąžinamas kiekvieni elemento funkcijos f reikšmių sąrašas (fe_1, ..., fe_n), kur fe_1 atspindi funkcijos f rezultatą, kai argumentas yra e_1.

topologinės instrukcijos

Formatas: (map f list)

Instrukcija map pakraunama į papildomą registrą F0. Naudojama funkcija yra pakraunama į papildomą registrą F1. Sąrašas pakraunamas į registrus $S_{0,0}$÷$S_{3,0}$. Kaip matyti iš pieš. 17a pagrindiniuose registruose saugomi elementai yra pastūmiami bazinių registrų matricoje per du žingsnius į viršų, t. y. registro $S_{x,0}$ turinys atsiranda registre $S_{x,2}$, kur x yra nuo 0 iki 3. Tai atliekama vertikalios magistralės linijų v0÷v3 pagalba. Kaip parodyta pieš. 17b papildomų registrų F0 ir F1 turinys horizontaliai kopijuojamas į bazinius registrus, t. y. F0 turinys kopijuojamas į registrus $S_{0,0}$÷$S_{3,0}$, o registro F1 turinys kopijuojamas į registrus $S_{0,1}$÷$S_{3,1}$. Jeigu elementas

yra ne sąrašas, jį turinčio registro turinys, pvz. $S_{1,2}$, ir žemiau esančio esančio registro, pvz. $S_{1,1}$, turinys pastumiami vienu žingsniu žemyn. Tokiu būdu funkcija atsiduria pagrindiniame registre, pvz. $S_{1,0}$, o jos elementas - registru žemiau, pvz. $S_{1,1}$. Jeigu elementas yra sąrašas, postūmis šiame registrų stulpelyje neatliekamas. Pieš. 17c laikoma, kad e_1, e_2 ir e_3 yra paprastos reikšmės, o e_4 - sąrašas. Kiekvienas bazinių registrų stulpelis yra išsaugomas kaip objektinės atminties išraiškos. Po to šios išraiškos vėl pakraunamos į bazinę ląstelę tolimesniam skaičiavimui. Jeigu išraiškoje yra tik paprasta reikšmė, ji normaliai pakraunama į bazinę ląstelę, pvz. f saugoma $S_{0,0}$ o ei saugoma $S_{1,0}$, akip parodyta pieš. 17d. Jeigu išraiškoje yra sąrašas, ji pakraunama, kaip aprašyta aukščiau (pieš. 17a), bet e_1 yra pirmas sąrašo e_4 elementas, e_2 -antras e_4 elementas ir t. t. Tai leidžia map instrukcijai dirbti su įstatytais sąrašais.

Tokiu būdu, dviejų lygių struktūros instrukcija map

$$(map, f, (e_1, .., e_n))$$

perrašoma į

$$((f, e_1), .., (f, e_n))$$

ir po įvykdymo perrašoma į vieno lygio struktūrą

$$(fe_1, .., fe_n)$$

kur fe_1 yra funkcijos f su argumentu e_1 rezultatas.

Instrukcija map, turinti trijų (arba daugiau) lygių struktūrą

$$(map, f, par(e_1, .., (e_k, .., e_m), .., e_n)),$$

kur $(e_k, .., e_m)$ yra įstatytas sąrašas, perrašoma į

$$par((f, e_1), .., (map, f, (e_k, .., e_m)), .., (f, e_n))$$

kaip tarpinį žingsnį ir po to

$$par((f, e_1), .., ((f, e_k), .., (f, e_m)), .., (f, e_n)),$$

kuri po įvykdymo perrašoma į dviejų lygių struktūrą

$$par(fe_1, .., (fe_k, .., fe_m), .., fe_n),$$

kur fe_1 yra funkcijos f su argumentu e_1 rezultatas, ir kur $(fe_k, .., fe_m)$ yra įstatytas sąrašas.

Tokiu būdu funkcija f rekursyviai panaudojama su visais argumentų sąrašo elementais.

Pavyzdys, kaip bazinė ląstelė pertvarko ir vykdo map instrukciją yra pateiktas žemiau. Yra naudojami sekantys sutrumpinimai: vietoje "registras" -reg, vietoje "identifikatorius" - ident ir vietoje "objektinė atmintis" - atmintis. Pavyzdžio instrukcija:

(map f (-1 -2 (-7 -8))),

kur f(x)=abs(x)+1. Mašinine kalba, naudojant mašininius identifikatorius tai atrodys kaip

id1 : (map f id2)

id2 : (-1 -2 id3)

id3 : (-7 -8)

kur identifikatorius id1 nurodo išraišką, turinčią (map f id2) struktūrą, ir t. t.

Žemiau i yra skaičiai nuo 0 iki 3. Atliekami sekantys žingsniai:

Žingsnis 1: map įrašoma reg F0, f - reg F1 ir ident id2 - reg $S_{0,0}$

Žingsnis 2: ident id2 išskleidžiamas, t. y. reg $S_{0,0}$ atsiranda -1, reg $S_{1,0}$ atsiranda -2, reg $S_{2,0}$ atsiranda ident id3

Žingsnis 3: reg $S_{i,0}$ turinys perduodamas į reg $S_{i,2}$. Nenaudojami registrai šiame žinsnyje nenaudojami

Žingsnis 4: map ir f kopijuojami horizontaliai, t. y. reg $S_{i,1}$ gauna f ir reg $S_{i,0}$ - map. Nenaudojami registrai šiame žinsnyje nenaudojami

Žingsnis 5: stulpeliai su esančia jų reg $S_{i,2}$ paprasta reikšme yra suspaudžiami vienu žingsniu žemyn, t. y. reg $S_{0,1}$ turi -1, reg $S_{0,0}$ turi f, reg $S_{1,1}$ turi -2 ir reg $S_{1,0}$ turi f; trečias stulpelis neliečiamas

Žingsnis 6: kiekvienas bazinių registrų matricos stulpelis išsaugomas atmintyje kaip:

id1: (id6 id7 id8)
id6: (f -1)
id7: (f-2)

id8: (map f id3)

Žingsnis 7: ident id6 apibrėžta išraiška pakraunama į pagrindinius registrus, f - į reg $S_{0,0}$ ir -1 - į reg $S_{1,0}$

Žingsnis 8: funkcija, t. y. f(x)=abs(x)+1, išskaičiuojama argumentams, rezultatas - 2 - išsaugomas reg $S_{0,0}$

Žingsnis 9: atliekama asociatyvinė paieška, ir visi surasti ident id6 pakeičiami 2:

id1: (2 id7 id8)
id7: (f -2)
id8: (map f id3)

Žingsnis 10: nuo 7 iki 9 žingsniai pakartojami ident id7 apskaičiavimui ir gaunama 3. Atmintis:

id1: (2 3 id8)
id8: (map f id3)

Žingsnis 11: žingsniai 1 iki 6 pakartojami ident id8 apskaičiavimui ir rezultate du bazinės matricos stulpeliai įrašomi į atmintį:

id1: (2 3 id8)
id8: (id9 id10)
id9: (f -7)
id10: (f -8)

Žingsnis 12: nuo 7 iki 9 žingsniai pakartojami ident id9 ir id10 apskaičiavimui ir gaunama atitinkamai 8 ir 9. Atmintis:

id1: (2 3 id8)
id8: (8 9)

Rezultatas yra: (2 3 (8 9)) - funkcija f buvo panaudota visiems argumentų sąrašo elementams.

Reikia pažymėti, kad tie patys rezultatai gali būti gauti skirtingu, žymiai efektyvesniu keliu. Pvz. užuot išsaugojus tarpinius rezultatus objektinėje atmintyje, iškart galima atlikti jų redukavimą/vykdymą.

transpose (tranponavimas)

Formatas: (transpose list(sąrašas))

Instrukcija transpose yra pakraunama į vieną iš papildomų registrų, pvz. F0, ir sąrašinis argumentas, pvz. sąrašų sąrašas, yra pakraunamas į bazinių registrų matricą (pieš. 18). Bazinių registrų matricos tyrinys yra transponuojamas. Taip, instrukcija transpose su trijų lygių struktūra

(transpose,
((e1,1, .., e1,m),
....
(en,1, .., en,m)))

yra įvykdoma ir jos rezultatas yra dviejų lygių struktūra

((e1,1, .., en,1),
....
(e1,m, .., en,m)))

Pavyzdys:

Sąrašinė struktūra:

((1 2 3 4),
(5 6 7 8),
(9 10 11 12)
(13 14 15 16)),

kur pirmasis sąrašas, t. y. (1 2 3 4), yra saugomas pirmame bazinių registrų stulpelyje, t. y. registruose $S_{0,0}$÷$S_{0,3}$, antrasis sąrašas, t. y. (5 6 7 8), yra saugomas antrame bazinių registrų stulpelyje, t. y. registruose $S_{1,0}$÷$S_{1,3}$, ir t. t., yra transponuojama į

((1 5 9 13),
(2 6 10 14),
(3 7 11 15)
(4 8 12 16)),

kur pirmasis sąrašas, t. y. (1 5 9 13), yra saugomas pirmame bazinių registrų stulpelyje, t. y. registruose $S_{0,0}$÷$S_{0,3}$, ir t. t.

swap (sukeitimas)

Formatas: (swap m list)

Instrukcija swap yra vykdoma tokiu būdu, kad instrukcija swap,

turinti trijų lygių struktūrą

(swap m (($e_{1,1}$, ..),
..
($e_{m,1}$, ..),
($e_{m+1,1}$, ..),
..
($e_{n,1}$, ..)))

kur sąrašų sąrašas su elementais $e_{i,j}$, kur i ir j žymi elemento poziciją bazinių registrų matricoje, yra perrašomas į dviejų lygių struktūrą

(($e_{1,1}$, ..),
..
($e_{m+1,1}$, ..),
($e_{m,1}$, ..),
..
($e_{n,1}$, ..))

taip kad elementas ($e_{m,1}$, ..) pasikeičia su elementu ($e_{m+1,1}$, ..) vietomis.

skip (praleisti)

Formatas: (skip m list)

Instrukcija skip yra vykdoma tokiu būdu, kad instrukcija skip, turinti trijų lygių struktūrą

(skip m (($e_{1,1}$, ..),
..
($e_{m-1,1}$, ..),
($e_{m,1}$, ..),
($e_{m+1,1}$, ..),
..
($e_{n,1}$, ..)))

kur sąrašų sąrašas su elementais $e_{i,j}$, kur i ir j žymi elemento poziciją bazinių registrų matricoje, yra perrašomas į dviejų lygių struktūrą

(($e_{1,1}$, ..),
..
($e_{m-1,1}$, ..),

$(e_{m+1,1}, ..),$
..
$(e_{n,1}, ..))$

taip kad sąrašas $(e_{m,1}, ..)$ yra pašalinamas.

4. Sąrašų išplėtimas

Sąrašas su uždaviniu yra pakraunamas į pagrindinius registrus. Jeigu sąraše yra elementų, kurie yra sąrašai, jie saugomi vertikaliai pagalbiniuose registruose.

Operacija expand_list atliekama per vieną taktą. Bazinių registrų turinys paslenkamas diagonaline kryptimi žemyn ir dešinėn per vieną žingsnį, nepaslinktas lieka pagrindinio registro turinys, kuris perkeliamas į vertikalią magistralę ir pakraunamas į šio stulpelio aukščiausią bazinį registrą (pieš. 19). Kartojant expand_list galima "užpildyti" pagrindinius registrus duomenimis.

BITINĖ LĄSTELĖ

Visa išradime pateikto redukuojančio procesoriaus atmintis yra numatyta padaryti pagal VLSI (Labai Didelės Integracijos) technologiją. Ir todėl kiekvienos bitinės ląstelės struktūra pritaikyta VLSI technologijai. Bitinės ląstelės grandinės įranga, kartu su draiveriais (valdymo mechanizmais), skirtais bitinės ląstelės linijoms d, d^* ir acc, yra parodyta pieš. 20. Bitinė ląstelė 15 yra apibrėžta brūkšnine linija.

Kiekviena bitinė ląstelė prie valdymo įrenginio yra prijungta dviejų linijų pagalba, pieš. 20 parodytų kaip linijos d ir d^*. Tokiu būdu informacinės magistralės dalis, einanti prie 38 bitų informacijos saugojimo elemento, susideda iš 76 laidų. Kiekviena tokia dalis prijunta prie atminties laukų 151, išdėstytų stulpeliais atminties ląstelėje.

Kaip matyti iš pieš. 20 bitinė ląstelė lastelė turi tik keturias jungtis, t. y. primoji jungtis V_{cc} yra kietai prijungta prie maitinimo įtampos, antroji, trečioji ir ketvirtoji jungtys acc, d, d^* yra nustatomos į vieną iš trijų skirtingų būsenų, kas bus aprašyta vėliau.

Pieš. 20 parodyta bitinė ląstelė yra keturių tranzistorių CMOS tipo statinė įrašoma ląstelė. Įrašymo kanalas yra MOP lauko tranzistorių I1 ir I2 su invertuojančiomis užtūromis

ištakas/santakas grandinė.Yra numatyta galimybė nustatyti varžinį įėjimą (neparodyta).

Ląstelė yra trigeriškai valdoma iš abiejų pusių. Tarp pasiekimo laido acc ir maitinimo laido V_{CC} lygiagrečiai prijungta dvi serijos grandinių, susidedančių iš atitinkamai MOP lauko tranzistorių T1, I1 ir T2, I2 ištakas/santakas ir įėjimo grandinių. Tranzistoriaus T1 santakas prijungtas prie tranzistoriaus T2 užtūros ir tranzistoriaus T2 santakas prijungtas prie tranzistoriaus T1 užtūros. Diodas D1 prijungtas tarp laido d ir sujungimo n1 tarp tranzistoriaus T1 santako, įėjimo I1 ir tranzistoriaus T2 užtūros. Diodas D2 prijungtas tarp laido d^* ir sujungimo n2 tarp tranzistoriaus T2 santako, įėjimo I2 ir tranzistoriaus T1 užtūros. Diodai D1 ir D2 realizuoti MOP lauko tranzistorių pagalba, kurių santakas ir užtūra sujungti tarpusavyje ir prijungti prie laido d ar d^* atitinkamai.

Esminis grandinės elementų scheminis sprendimas yra tas, kad diodai D1 ir D2 leidžia tekėti srovei tik viena kryptimi d ir d^* laidų atžvilgiu, ir kad tranzistoriai T1 ir T2 yra aktyvūs elementai, per kuriuos tekanti srovė gali būti valdoma užtūrų potencialo pagalba. Sujungimai n1 ir n2 yra mazgai, kuriuose kaip potencialo lygis atsispindi vienas saugomos informacijos bitas. Kiekvienas įėjimas I1 ir I2 elgiasi kaip varžinis elementas.

Pieš. 20 yra parodyta V_{CC} su aukštu potencialo lygiu. Diodų D1 ir D2 pagalba srovė teka atitinkamai iš laido d ar d^* į mazgus n1 ar n2. Kai tranzistorių T1 ar T2 užtūros potencialas padidėja, jų varža sumažėja ir mazguose sumažėja potencialas. Vienok galimi ir kitokie scheminiai sprendimai, kur srovė gali tekėti priešinga kryptimi, negu parodyta pieš. 20.

Bitinė ląstelė gali saugoti reikšmę v_{store}, kuri gali būti 'teisinga' arba 'neteisinga'. Bitinės ląstelės strktūros būsenos gali būti valdomos linijų acc, d ir d^* potencialais.

Valdymo būsenos gali būti aukštas lygis, žemas lygis - visuose laiduose, srovė teka į ląstelę ir srovė teka iš ląstelės - laide acc. Laidas acc yra pasiekimo laidas, einantis iš viršūnės 16 ir prijungtas prie visų atminties lauko bitinių ląstelių 15. Trečios ir ketvirtos linijų d ir d^* signalai yra invertuoti vienas kito atžvilgiu, kai į ląstelę yra įrašoma arba iš jos nuskaitoma, esant

pasiekimo linijoje acc signalui LOW.

Elemento viršūnės 16 valdymo grandinė ir prasmės stiprintuvas

Elemento viršūnės 16 valdymo grandinė ir prasmės stiprintuvas parodytas brūkšniuotame stačiakampyje pieš. 20. Pirmojo tranzistoriaus T3 ištakas prijungtas prie įtampos V_r, santakas - prie atminties ląstelės lauko 151 visų bitinių ląstelių 15 pasiekimo laidų acc ir užtūra - prie taktinio signalo generatoriaus signalo prech. Sekančio tranzistoriaus T4 ištakas prijungtas prie įtampos 0V, santakas - prie atminties ląstelės lauko 151 visų bitinių ląstelių 15 pasiekimo laidų acc ir užtūra - prie įtampos V3, kuri yra aukšto lygio, kai pasiekimo laide acc yra užstatoma įtampa 0V. Kaip pažymėta aukščiau, laidas acc prijungtas prie visų išraiškos elemento bitinių ląstelių, pvz. 38 bitinių ląstelių. Laido acc įtampa stiprinama stiprintuve AMP.

Interfeiso 7 valdymo grandinė ir prasmės stiprintuvas

Interfeiso 7 valdymo grandinė ir prasmės stiprintuvas bitinės ląstelės laidams d ir d^* yra parodyti kitame brūkšniuotame stačiakampyje pieš. 20. Vienok reikia pažymėti, kad čia parodyta tik vienas iš galimų schemotechnių linijų d ir d^* aptarnavimo realizavimo variantų. Įėjimas/išėjimas IN/OUT prijungtas prie pieš. 1 parodytos bazinės ląstelės 2. Ši grandinė yra viena iš daugelio panašių grandinių, kurios gali funkcionuoti kaip interfeisas 7 tarp objektinės atminties 1 ir bazinės ląstelės 2.

Laido d įrašymo grandinė įjungia pirmąją tranzistorių T5 ir T6 porą, iš kurių pirmasis yra n-tipo, o antrasis p-tipo, ir kurių santakai prijungti prie laido d, tokiu būdu sudarant įtampos daliklį. Tranzistoriaus T5 ištakas prijungtas prie potencialo Vr, o jo užtūra gauna paruošiamąjį užkrovimo (paruošimo) signalą prech. Kito tranzistoriaus T6 ištakas prijungtas prie potencialo Vcc, o jo užtūra gauna valdymo signalą V4, kuris numetamas į žemą lygį, kai potencialas Vcc paduodamas į laidą d. Laido d įrašymo grandinės p-tipo tranzistoriaus T9 ir n-tipo tranzistoriaus T10 ištakas/santakas grandinė prijungta tarp įtampos šaltinio Vcc ir n-tipo tranzistoriaus T11, kurio ištakas prijungtas prie žemės, o užtūra - prie įrašymo įėjimo, prijungto prie išorinio valdymo, santako. Tranzistorių T9 ir T10 sujungti santakai prijungti prie tranzistoriaus T6 užtūros su įtampa V4. Tranzistoriaus T9 užtūra gauna

invertuotą paruošimo signalą prech*, tokiu būdu paruošimo fazės metu per atvirą tranzistorių T9 prijungiant tranzistorių T6 prie šaltinio įtampos Vcc.

Laido d* įrašymo grandinė įjungia kitą tranzistorių T7 ir T8 porą, iš kurių pirmasis yra n-tipo, o antrasis p-tipo, ir kurių santakai prijungti prie laido d*, tokiu būdu sudarant įtampos daliklį. Tranzistoriaus T7 ištakas prijungtas prie potencialo Vr, o jo užtūra gauna paruošiamąjį užkrovimo (paruošimo) signalą prech. Kito tranzistoriaus T8 ištakas prijungtas prie potencialo Vcc, o jo užtūra gauna valdymo signalą V5, kuris numetamas į žemą lygį, kai potencialas Vcc paduodamas į laidą d*.

Laido d* įrašymo grandinės p-tipo tranzistoriaus T12 ir n-tipo tranzistoriaus T13 ištakas/santakas grandinė prijungta tarp įtampos šaltinio Vcc ir n-tipo tranzistoriaus T11 santako. Tranzistorių T12 ir T13 sujungti santakai prijungti prie tranzistoriaus T8 užtūros su įtampa V5. Tranzistoriaus T12 užtūra gauna invertuotą paruošimo signalą prech* tokiu būdu paruošimo fazės metu per atvirą tranzistorių T12 prijungiant tranzistorių T8 prie šaltinio įtampos Vcc.

Išorinė linija IN/OUT, skirta įėjimui ir išėjimui yra prijungta prie dviejų trijų būsenų invertorių. Vienas iš trijų būsenų invertorių, kurio išėjimas prijungtas prie IN/OUT linijos, įjungia dviejų n-tipo tranzistorių T14, T15 ir dviejų p-tipo tranzistorių T16, T17 ištakas/santakas grandinių seriją. Tranzistoriaus T16 užtūra prijungta prie valdymo laido su signalu bitin ir tranzistoriaus T15 užtūra gauna invertuotą signalą bitin*. Kitas iš trijų būsenų invertorių, kurio įėjimas prijungtas prie IN/OUT linijos, įjungia dviejų n-tipo tranzistorių T18, T19 ir dviejų p-tipo tranzistorių T20, T21 ištakas/santakas grandinių seriją. Tranzistoriaus T19 užtūra prijungta prie valdymo laido su signalu bitin ir tranzistoriaus T15 užtūra gauna invertuotą signalą bitin*. Šio trijų būsenų invertoriaus išėjimas prijungtas prie tranzistoriaus T13 užtūros ir per invertorių INV prie tranzistoriaus T10 užtūros.

Skaitymo stiprintuvas įjungia n-tipo tranzistorių T22, kurio ištakas prijungtas prie žemės, užtūra prijungta prie pastovios įtampos Vbias, kuri laiko tranzistorių T22 pastoviai atidarytą ir

priverčia jį funkcionuoti kaip srovės generatorių, ir santakas prijungtas prie n-tipo tranzistorių T23 ir T24 ir p-tipo tranzistorių T24 ir T25 lygiagrečiai sujungtų ištakas/santakas grandinių, kurių kitas galas prijungtas prie šaltinio įtampos Vcc. p-tipo tranzistorių T24 ir T25 užtūros sujungtos tarpusavyje ir prijungtos prie tranzistorių T23 ir T24 sujungtų santakų. Tranzistoriaus T23 užtūra yra prijungta prie linijos d, o tranzistoriaus T25 užtūra yra prijungta prie linijos d*.

Kiekvieno taktinio signalo metu generuojami signalai prech ir prech* yra skirstomi į paruošimo fazę, kada signalas prech yra aukštas, ir vykdymo fazę, kada prech signalas yra žemas ir kitų valdančiųjų signalų pagalba nustatoma operacija, kuri tuti būti vykdoma. Paruošimo fazės metu linijose d, d* ir acc per atitinkamus tranzistorius T5, T7 ir T3 yra užstatoma įtampa Vr.

Signalų bitin ir bitin* pagalba kontroliuojamas duomenų siuntimas iš ląstelės 15 arba iš jos. Kai signalas bitin yra žemas o signalas bitin* yra aukštas, duomenys pirmojo trijų būsenų invertoriaus pagalba yra siunčiami iš ląstelės į liniją IN/OUT. Kai signalas bitin yra aukštas o signalas bitin* yra žemas, duomenys antrojo trijų būsenų invertoriaus pagalba yra gaunami į ląstelę iš linijos IN/OUT.

Po paruošimo fazės, kai linijose d, d* ir acc yra įtampa Vr, prasideda antroji skaitymo operacijos fazė, kurios metu linijose d ir d* yra kintanti įtampos reikšmė, o linijoje acc, įtampos V3 pagalba atidarius tranzistorių T4, užstatoma 0V įtampa. Tokiu būdu mazgo, pvz. n1, potencialas įgauna mažesnę reikšmę tarp Vr ir 0V. To pasekmėje srovė teka iš linijos d į mazgą n1 ir liniją acc. Ši srovė nukrauna liniją d, t. y. linijos d įtampa krenta. Šis įtampos sumažėjimas yra fiksuojamas skaitymo stiprintuvo tranzistoriuose T22÷T26. Šis rezultatas yra perduodamas į tranzistorių T25 ir T26 santakų sujungimą ir toliau paduodamas į pirmojo trijų būsenų invertoriaus, susidedančio iš tranzistorių T14÷T17, įėjimą. Kai signalas bitin yra žemas, o signalas bitin* yra aukštas, perduodama skaitoma bitinė reikšmė į įėjimo/išėjimo liniją IN/OUT. Yra svarbu pažymėti, kad antros fazės metu linijų d ir d* reikšmės nenaudojamos.

Pradžioje skaitymo opercijos metu linijose d ir d* yra įtampa Vr.

Nors jose abiejose ir palaikomas potencialas Vr, bet po kurio laiko vienoje iš jų jis dėl įėjimo srovės krenta. Kadangi potencialas yra apbrėžiamas kaip "žemas", tai žemas potencialas yra žemesnis už "žemą" potencialą. d ir d* atsispindi skaitomos reikšmės. d mažesnis už d* duoda "neteisinga", d didesnis už d* duoda "teisinga". Operacijose: nerašyti, įrašyti teisinga, įrašyti neteisinga, nerašyti ir nelyginti linijų d ir d* potencialai yra neinformatyvūs.

Po paruošimo fazės, kai linijose d, d* ir acc yra įtampa Vr, prasideda antroji skaitymo operacijos fazė, linija acc , įtampos V3 pagalba atidarius tranzistorių T4, yra numetama į 0V. Reikšmė, kuri turi būti saugoma, yra paduodama į įėjimo/išėjimo liniją IN/OUT. Kai signalas bitin yra aukštas o signalas bitin* yra žemas, antrasis trijų būsenų invertorius, susidedantis iš tranzistorių T18÷T21, perduoda linijos IN/OUT reikšmę į savo išėjimą. Tranzistoriaus T11 užtūroje esantys aukšto lygio valdantieji signalai, šio tranzistoriaus pagalba prijungia 0V prie tranzistorių T10 ir T13 ištakų.

Antrojo trijų būsenų invertoriaus (tranzistoriai T18÷T21) generuojamas aukšto lygio signalas, t. y. rašomas "0" arba 'neteisinga', permeta tranzistorių T13 į atidarytą būseną, pažemindamas įtampą V5, atidaro tranzistorių T8 ir užstato linijoje d* aukštą įtampos lygį Vcc. Invertuotas antrojo trijų būsenų invertoriaus žemo lygio išėjimo signalas, paduotas į tranzistoriaus T10 užtūrą, laiko jį uždarytu, ir tokiu būdu įtampa V4, paruošimo fazės metu pakilusi iki Vcc lygio, jį išlaiko. Tranzistorius T6 yra uždarytas, ir įtampa Vr, paruošimo fazės metu per tranzistorių T5 prijungta prie linijos d, išlaiko savo dydį.

Antrojo trijų būsenų invertoriaus (tranzistoriai T18÷T21) generuojamas žemo lygio signalas, t. y. rašomas "1" arba 'teisinga', valdo įrašymo grandinę T5, T6, T9, T10, kurios pagalba linija d per invertorių INV gauna įtampos Vcc aukšto lygio reikšmę, ir šis signalas tuo pat metu įrašymo grandinės T7, T8, T12, T13 pagalba išlaiko linijoje d*gautą paruošimo fazės metu įtampą Vr.

Kaip matyti iš pavyzdžių, aprašytų aukščiau, pieš. 20 parodytoje įrangoje mazgai n1 ir n2 naudojami sekančiai: vienas arba abu mazgai n1 ir n2 nukraunami arba pakraunami operacijos ciklo

antrosios fazės metu priklausomai nuo to, koks valdymo signalas ateina - V3, V4 ar V5, t. y. ar linija acc numetama į 0V, ar vienas ar abu laidai d ir d* gauna Vcc.

Kaip paminėta aukščiau, kiekvienas operacijos ciklas susideda iš paruošimo fazės ir vykdymo fazės. Kai linijoje acc yra aukštas lygis, tai reiškia kad signalas V3 nevaldo tranzistoriaus T4, ir tokiu būdu tranzistorius neužstato 0V įtampos linijoje acc vykdymo metu. Atitinkamai, laiduose d arba d* esantis žemas lygis reiškia, kad signalai V5 arba V6 per atitinkamus tranzistorius T6 arba T8 neprijungia šių linijų prie įtampos Vcc, kuri yra aukštesnė už Vr, vykdymo metu. Kai linijose d arba d* yra aukštas lygis, tai reiškia, tranzistoriai gauna valdymo signalą prijungti prie linijos įtampą Vcc.

Atminties dydis gali būti didelis, pvz. gali įjungti 256 atminties ląsteles, kas reiškia, kad kiekviena tranzistorių - T6, T7 ir T7, T8 - pora, yra prijungta prie visų atminties ląstelių, kaip pvz. prie 256, vienos bitinės ląstelės. Atitinkamai, tranzistoriai turi būti parinkti atsižvelgiant į magistralės talpumą ir reikalingą greitį.

Įtampa Vr gali būti nustatyta užtrumpinto invertoriaus pagalba, tam kad išlaikyti žinomą ryšį tarp Vr ir stiprintuvo invertoriaus. Viršūnės pasiekimo grandinės turi valdyti ir gauti informaciją iš bitinių ląstelių.

Sekančios funkcinės būsenos nustatomos valdymo signalų pagalba:

rest (ilsėtis)	ląstelė saugo reikšmę vstore
read false (nuskaityti 'neteisinga')	reikšmė vstore='neteisinga' yra nuskaitoma
read true (nuskaityti 'teisinga')	reikšmė vstore='teisinga' yra nuskaitoma
don't read (nenuskaityti)	ląstelė saugo reikšmę vstore
write false (įrašyti 'neteisinga')	saugoma reikšmė vstore yra nustatoma į 'neteisinga'

write true (įrašyti 'teisinga')	saugoma reikšmė vstore yra nustatoma į 'teisinga'
don't write (nerašyti)	ląstelė saugo reikšmę vstore
comp. false (sulyginti su 'neteisinga')	saugoma reikšmė vstore yra sulyginama su reikšme 'neteisinga'
comp. true (sulyginti su 'teisinga')	saugoma reikšmė vstore yra sulyginama su reikšme 'teisinga'
don't comp. (nesulyginti)	ląstelė saugo reikšmę vstore.

Toliau pateikta bitinės ląstelės skirtingų operacijų režimų lentelė:

Op. režimas	acc	d	d*
rest	žemas	žemas	žemas
read false	žemas	srovė įteka	aukštas
read true	žemas	aukštas	srovė įteka
don't read	aukštas	pasirenkamas	pasirenkamas
write false	žemas	žemas	aukštas
write true	žemas	aukštas	žemas
don't write	aukštas	pasirenkamas	pasirenkamas
comp. false	pasirenkamas	žemas	aukštas
comp. true	pasirenkamas	aukštas	žemas
don't comp.	pasirenkamas	žemas	žemas

Operacijose comp. false ir comp. true linija acc turi būseną 'srovė išteka' kai sulyginimo rezultatas yra DIFFERENT (SKIRTINGA).

Operacijose comp. false ir comp. true linijoje acc (pasiekimo linijoje) pasirodo sulyginimo rezultatas. Linija acc gauna Vr

reikšmę, duomenys paduodami linija d, o jų invertuota reikšmė - linija d*. Jeigu bitinės ląstelės turinys nesutampa su įėjimo duomenimis, linija acc yra pakraunama per diodus D1 arba D2 bei atitinkamus n-tipo tranzistorius T1 arba T2. Tai aptinka viršūnės 16 stiprintuvas AMP. Jeigu sulyginime gauta FIT (SUTAMPA), linija acc išlaiko potencialą Vr.

Srovė įteka ir srovė išteka reiškia, kad tam tikra linija tam tikru laiko momentu pakraunama arba iškraunama elektros krūviais. Tai daroma, pradiniu momentu nustatant linijoje HIGH (AUKŠTAS) arba LOW (ŽEMAS), esant operacijos režimui REST (RAMYBĖ) ir vėliau pakeičiant esamu režimu. Srovė atitinkamai nukrauna arba užkrauna tam tikrą liniją. Jeigu srovė neteka, krūviai yra nepernešami. Tokiu būdu, tam tikrą laiko tarpą įtampa lieka pastovi.

OBJEKTINĖS ATMINTIES VALDYMO GRANDINĖS

Prioritetų dekoderis (pieš. 21A ir B)

Prioritetų dekoderio įranga, parodyta pieš. 21A ir 21B yra sudalinta į keturis blokus. Kaip parodyta pieš. 21A, kiekvienas blokas kairėje turi porą laidų granta ir reqa ir dešinėje laidus req0, grant0, ..., req3, grant3.

Kaip parodyta pieš. 21B, kiekvienas bloko 520 dešnieji laidai prijungti prie kiekvieno iš keturių blokų (parodyti išoriniai blokai 521 ir 522) kairiųjų laidų. Blokai 520, 522 ir blokai 521, 523 yra sujungti inveruojančiais stiprintuvais 523 ir 4524, kur stiprintuvas 523 perduoda informaciją žemesniam blokų grandinėlės blokui, kurio prioriteto reikalavimas yra reikalingas aukštesniam blokų grandinėlės blokui, ir stiprintuvas 524 perduoda informaciją aukštesniam blokų grandinėlės blokui, negu tam kuriam yra duotas leidimas. Blokų 521, .., 522 skaičius sekančiame 4-blokų stulpelyje yra keturi.

Kiekvienas antrojo stulpelio blokas prijungtas prie keturių blokų ketvertuko trečiajame stulpelyje taip pat, kaip blokai antrajame stulpelyje prijungti prie 4-bloko 520. 4-blokų skaičius trečiajame stulpelyje bus lygus šešiolikai. Parodyti išoriniai 4-blokai 525 ir 526.

Kiekvienas trečiojo stulpelio blokas prijungtas prie keturių blokų ketvertuko ketvirtame stulpelyje minėtu būdu. 4-blokų

skaičius ketvirtajame stulpelyje bus lygus šešiasdešimt keturi. Parodyti tiktai išoriniai 4-blokai 527 ir 528.

Dešinieji kiekvieno ketvirtojo stulpelio 4-bloko laidai prijungti prie objektinės atminties. Kaip matyti iš pieš. 21B, kiekviena laidų pora tarnauja kaip atminties ląstelės magistralės 18 ir 19.

Aštuniasdešimt penki blokai aprūpina 256 išraiškas. Žemiausias blokas 528 aptarnauja žemiausias ląsteles (žemyn iki ląstelės 0), aukščiausias blokas 527 aptarnauja aukščiausias ląsteles (aukštyn iki ląstelės 255).

Pieš. 21A ir 21B parodyta sistema naudoja domino pakraunamą logiką, kada pilnas prioritetų dekoderis įjungia kaskadinius domino laipsnius, atitinkamai skirtus perduoti pareikalavimo signalui, pvz. req0 iš žemiausio bloko 528, per visus prioritetų dekoderio blokus ir atgal, grąžinant 'neteisinga' leidimo signalą visoms atminties ląstelėms, išskyrus atminties ląstelę 0.

Kaip parodyta pieš. 21A, kievienam bloke yra 5 eilės MOP lauko tranzistorių, ir eilė, esanti aukščiau įjungia vienu MOP lauko tranzistoriu daugiau negu žemesnė eilė, išskyrus penktąją eilę, kurios tranzistorių skaičius sutampa su ketvirtosios.

Kiekvienas dešiniausias MOP lauko tranzistorius pirmose apatinėse eilėse (t. y. tranzistoriai $Tr_{0,0} \div Tr_{3,0}$), yra p-tipo, jo užtūra prijungta prie taktinio signalo, santakas prijungtas prie pliusinės maitinimo įtampos ir ištakas prijungtas prie atitikamų leidimo laidų grant0, grant1, grant2 arba grant3. MOP lauko tranzistoriaus $Tr_{4,0}$, esančio viršutinėje eilėje, užtūra, skirtingai negu tranzistorių $Tr_{0,0} \div Tr_{3,0}$ užtūros, per invertorių 523 yra prijungta prie linijos reqa sekančiame žemesniame bloke.

Kitų n-tipo MOP lauko tranzistorių, $Tr_{0,1}$ pirmoje eilėje, $Tr_{1,1}$, $Tr_{1,2}$, antroje eilėje, $Tr_{2,1}$, $Tr_{2,2}$, $Tr_{2,3}$ trečioje eilėje, $Tr_{3,1}$, $Tr_{3,2}$, $Tr_{3,3}$, $Tr_{3,4}$ ketvirtoje eilėje, $Tr_{4,1}$, $Tr_{4,2}$, $Tr_{4,3}$, $Tr_{4,4}$, $Tr_{4,5}$ penktoje eilėje, ištakai prijungti prie žemės ir santakai prie atitinkamų linijų grant0, grant1, grant2, grant3 (apatinėse eilėse) ir linijos reqa (viršutinėje eilėje).

Linija granta yra prijungta prie kiekvienos MOP lauko tranzistorių $Tr_{0,1}$, $Tr_{1,1}$, $Tr_{2,1}$, $Tr_{3,1}$ užtūros. Linija req0 yra prijungta prie kiekvienos MOP lauko tranzistorių $Tr_{1,2}$, $Tr_{2,2}$,

$Tr_{3,2}$, $Tr_{4,2}$ užtūros. Linija req1 yra prijungta prie kiekvienos MOP lauko tranzistorių $Tr_{2,3}$, $Tr_{3,3}$, $Tr_{4,3}$ užtūros. Linija req2 yra prijungta prie kiekvienos MOP lauko $Tr_{3,4}$, $Tr_{4,4}$ užtūros.

Prioritetų dekoderio darbas susideda iš dviejų fazių. Pirmos fazės metu, kai taktinis signalas yra žemo lygio, visi granti padaromo aukšto lygio ('teisinga'). Tada viso reqi signalai bus žemo lygio (nėra reikalavimo). Sekančios fazės metu pakrovimas baigiasi, t. y. taktinis signalas tampa aukšto lygio. Tada vienas ar keli reqi išėjimai pakils iki aukšto lygio, kas numes visus granti į žemiausią lygį (neparodyta) nustatys reqa į žemą lygį. Kai reqa taps žemas, kaimyninio 4-bloko kairėje reqi pieš. 21 gaus aukštą lygį. 4-bloko 520 signalai reqa ir granta, kuris yra prijungtas prie žemės, netruri įtakos. 4-bloko 520 signalas reqa linijoje 14 (pieš. 2 ir 3) išduoda rezultatą "ANY", kadangi jis numetamas žemyn, kai bet kuris ląstelės išraiškos reqi tampa aukštu.

Elemento viršūnė (pieš. 22)

Smulki elemento viršūnės 16 schema parodyta pieš. 22. Elemento viršūnė 16 valdo atminties ląstelių 10 pasiekimo liniją acc, ji nustato pasiekimo liniją acc ir liniją ANY 14, čia vadinama any_type, ji vykdo operacijas wired_and ir wired_nor magistralėse a ir b, ir ji nuskaito magistralių a ir b reikšmes. Ji įjungia vidinį dinaminės atminties bitą.

n-kanalo MOP lauko tranzistoriaus n0 ištakas/santakas grandinė prijungta tarp įtampos Vr (tokia pati, kaip įtampa Vr pieš. 20) ir pasiekimo linijos acc. Taktinis dažnis cpb patenka į MOP lauko tranzistoriaus n0 užtūrą. n-kanalo MOP lauko tranzistoriaus n1 ištakas/santakas grandinė prijungta tarp žemės ir pasiekimo linijos acc.

Dviejų p-kanalo MOP lauko tranzistorių p2, p3 ištakas/santakas, p-kanalo MOP lauko tranzistoriaus p4 ištakas/santakas ir n-kanalo MOP lauko tranzistorių n5 santakas/ištakas grandinės lygiagrečiai yra prijungtos tarp maitinimo įtampos Vcc ir žemės.

Linija any_type, tiesiogiai prijungta prie artimiausios bitinės ląstelės, yra prijungta prie MOP lauko tranzistoriaus p2 užtūros. Centrinio valdymo įrenginio linija match yra prijungta prie MOP

lauko tranzistoriaus p3 užtūros. Pasiekimo linija acc per invertorių INV1, įjungiantį sujungtas p-kanalo MOP lauko tranzistoriaus p1 ir n-kanalo MOP lauko tranzistoriaus n6 ištakas/santakas grandines ir kurių užtūros prijungtos prie pasiekimo linijos, yra prijungta prie MOP lauko tranzistoriaus p4 užtūros. n-kanalo MOP lauko tranzistoriaus n7 ištakas/santakas grandinė įjungta tarp invertoriaus INV1 ir žemės. Linija eval.s, t. y. išrinkimo įvertinimas, einanti nuo centrinio valdymo įrenginio yra prijungta prie MOP lauko tranzistoriaus n7 užtūros. Linija set.s, t. y. nustatyti išrinkimą, einanti nuo centrinio valdymo įrenginio yra prijungta prie MOP lauko tranzistoriaus n5 užtūros.

Sujungimo taškas i1, vadinamas išrinkimo mazgu, esantis tarp MOP lauko tranzistorių p4 ir n5 santakų, yra prijungtas prie p-kanalo MOP lauko tranzistoriaus p6, kurio ištakas per p-kanalo MOP lauko tranzistoriaus p7 santakas/ištakas grandinę yra prijungtas prie maitinimo įtampos +Vcc. Linija reset.b, einanti nuo centrinio valdymo įrenginio, yra prijungta prie MOP lauko tranzistoriaus p6 užtūros. Linija b, kuri yra identiška linijai b iš pieš. 3, yra prijungta prie MOP lauko tranzistoriaus p7 užtūros.

p-kanalo MOP lauko tranzistoriaus p8 santakas prijungtas prie linijos a, kuri yra ta pati kaip ir pieš. 3, o jo ištakas prijungtas prie sujungimo i2. Linija Wand.a, t. y. prijungtas IR a, einanti nuo centrinio valdymo įrenginio, yra prijungta prie invertuotos MOP lauko tranzistoriaus p8 užtūros. p-kanalo MOP lauko tranzistoriaus p9 santakas prijungtas prie linijos b, o jo ištakas - prie sujungimo i2. Linija Wand.b, t. y. prijungtas IR b, einanti nuo centrinio valdymo įrenginio, yra prijungta prie invertuotos MOP lauko tranzistoriaus p9 užtūros. p-kanalo MOP lauko tranzistoriaus p10 santakas prie sujungimo taško i2 o jo ištakas - prie maitinimo įtampos +Vcc. Sujungimo taškas i1 per invertorių INV2 prijungtas prie MOP lauko tranzistoriaus p10 invertuotos užtūros. Invertorius INV2 įjungia lygiagrečiai sujungtas p-kanalo MOP lauko tranzistoriaus p14 ir n-kanalo MOP lauko tranzistoriaus n8 ištakas/santakas grandines, įjungtas tarp maitinimo įtampos +Vcc ir žemės, šių tranzistorių užtūros prijungtos prie sujungimo taško i1.

Dviejų p-kanalo MOP lauko tranzistorių p22 ir p12 santakas/ištakas grandinės yra prijungtos tarp linijos a ir maitinimo

įtampos +Vcc. Linija wor, t. y. prijungtas ARBA, einanti nuo centrinio valdymo įrenginio, yra prijungta prie MOP lauko tranzistoriaus p12 užtūros. Sujungimo taškas i1 yr prijungtas prie MOP lauko tranzistoriaus p11 užtūros. p-kanalo MOP lauko tranzistoriaus p13 ištakas/santakas prijungtas tarp linijos a ir sujungimo taško i1. Linija s.a, t. y. išrinkti a, einanti nuo centrinio valdymo įrenginio, yra prijungta prie MOP lauko tranzistoriaus p13 užtūros.

n-kanalo MOP lauko tranzistoriaus n2, p-kanalo MOP lauko tranzistoriaus p15, p-kanalo MOP lauko tranzistoriaus p16 santakas/ištakas grandinės įjungtos tarp žemės ir maitinimo įtampos +Vcc. Sujungimo taškas tarp MOP lauko tranzistorių n2 ir p15 ištakų yra prijungtas prie MOP lauko tranzistoriaus n1 užtūros. p-kanalo MOP lauko tranzistoriai p17 ir p18 yra įjungti tarp sujungimo taško i3 ir maitinimo įtampos +Vcc. Sujungimo taškas i1 prijungtas prie MOP lauko tranzistoriaus p18 užtūros. Linija r/w.b, t. y. nuskaityti/įrašyti b, einanti nuo centrinio valdymo įrenginio, yra prijungta prie invertuotos MOP lauko tranzistoriaus p15 užtūros. Linija r/w.s, t. y. nuskaityti/įrašyti išrinkimą, einanti nuo centrinio valdymo įrenginio, yra prijungta prie MOP lauko tranzistoriaus p17 užtūros. Linija r/w.r, t. y. nuskaitymo/įrašymo numetimas (naudojamas numesti taško i3 reikšmei po nuskaitymo ar įrašymo operacijos), einanti nuo centrinio valdymo įrenginio, yra prijungta prie MOP lauko tranzistoriaus n2 užtūros.

Pieš. 22 parodyto elemento viršūnės funkcionavimo logika yra sekanti. MOP lauko tranzistorius n0 pakrauna pasiekimo liniją acc kiekvieno neigiamo taktinio impulso metu, ir MOP lauko tranzistorius n1 nustato joje žemą lygį įrašymo ar skaitymo metu. MOP lauko tranzistorius numeta tašką i3 į žemą lygį ir tokiu būdu laukimo režime MOP lauko tranzistorius n1 būna uždarytas.

MOP lauko tranzistoriai vykdo valdomas išrinkimo mazgo pagalba nuskaitymo/įrašymo operacijas. Tai naudojama, pvz. instrukcijoje match, kai ištrinama ląstelės išraiškos žymė arba nustatant ląstelės išraišką neužstatant žymės, kai vykdomos operacijos priklauso nuo magistralių a ir b reikšmių.

Funkcija match sulygina reikšmę, pvz. taip vadinamą uždavinio reikšmę bazinėje ląstelėje ir atminties ląstelėje apibrėžiančią dvi sekas. mach rezultatas bus 'neteisinga', jeigu šios dvi sekos

nepersikerta. Šiuo atveju įertinama ir tai, kad išraiškų dalys gali būti neapibrėžtos.

MOP lauko tranzistorius n5 pakrauna išrinkimo tašką i1, ir MOP lauko tranzistoriai p2÷p4 valdymo įrenginio 6 pagalba nustato tą reikšmę MOP lauko tranzistoriaus n7 užtūroje. Priklausomai nuo valdymo įrenginio signalų MOP lauko tranzistoriai p8÷p10 vykdo wired_and operaciją magistralėse a ir/arba b. Esant valdymo įrenginio signalui MOP lauko tranzistoriai p11 ri p12 vykdo wired_or operaciją magistralėje a .

Invertoriaus INV2 pagalba invertuojamas išrinkimo taške esantis išrinkimo bitas. Tai reikalinga operacijų wired_or ir wired_and vykdymui. MOP lauko tranzistorius p13 perduoda magistralėje a esančią reikšmę į išrinkimo tašką. Perduodamas tik aukštas lygis, kadangi šis taškas yra žemame lygyje. MOP lauko tranzistoriai p6 ir p7 naudojami išrinkti aukštą lygį kai tai yra reikalinga. Tai naudojama instrukcijose, kurios nuskaito pažymėtą atminties ląstelę ir nuvalo jos žymę, kadangi išrinkimo taškas turi būti numestas iš karto po nuskaitymo operacijos. Ši savybė taip pat naudojama ir kitų tipų instrukcijose, kai reikalinga atlikti loginę IR operaciją.

Signalas any_type yra tiesiai prijungtas prie artimiausios bitinės ląstelės. Jis atspindi saugomos reikšmės tipą. match (atitikti) operacijos metu MOP lauko tranzistorius p3 yra uždaromas, ir tokiu būdu aukšta linijos, kurioje yra signalas any_type, reikšmė duoda match 'teisinga'. Panašiai, tikrinant kurios ląstelės yra neužimtos, aukštas linijos su signalu any_type lygis, kai MOP lauko tranzistorius p3 yra uždarytas, iššaukia tai, kad išrinkimo signalo lygis išrinkimo taške lieka žemas. Ekvivalentiškumo testų vykdymo metu MOP lauko tranzistorius p3 yra atidaromas.

Išraiškos viršūnė (pieš. 23)

Parodyta pieš. 23 išraiškos viršūnės 17 įranga leidžia vykdyti prioritetų ir MODE (REŽIMAS) operacijas a arba b magistralėse. Abi operacijos gali būti vykdomos vienu metu skirtingose magistralėse. MODE operacija, t. y. grandinės iš pieš. 2 ir 3 magistralės 12 (MODE) operacija, reaguoja į magistralės a lygį (ir aukštą, ir žemą), ir reaguoja tik į žemą b magistralės lygį. Išraiškos viršūnė

taip pat gali numesti magistralės 13 (pieš. 2) globalinį signalą MORE, ir nuskaityti bei įrašyti senus magistralės cuomenis į magistrales a ir b.

Abi grandinės, skirtos prioritetų ir MODE operacijoms vykdyti, yra realizuotos pagal dviejų laipsnių domino principą. Pirmajame laipsnyje nustatoma, kuri iš magistralių bus įėjimas ir sutvarkomas poliariškumas, o antrasis laipsnis vykdo pačią operaciją. Išraiškos viršūnė darbas skirstomas į dvi fazes, iš kurių pirmoji yra paruošiamojo pakrovimo fazė, o antroji vykkdymo fazė. Tam naudojami n-kanalo MOP lauko tranzistoriai n20, n21, n22 ir n23, kurių užtūros valdomos centinio valdymo įrenginio taktiniu signalu prech. n20 užkrauna liniją a, n21 užkrauna liniją b, n22 užkrauna liniją req, einančią į prioritetų dekoderį 11 ir n23 numeta sujungimo tinklą F1 į žemą lygį.

Sujungimo tinklas F1 susideda iš dviejų lygiagrečių p-kanalo MOP lauko tranzistorių p20, p21 ir p22, p23 porų, su sujungtomis ištakas/santakas grandinėmis. Tinklas F1 viename gale per MOP lauko tranzistoriaus p23 ištakas/santakas grandinę yra prijungtas prie žemės ir kitame gale gauna maitinimom įtampą +Vcc. Magistralė a prijungta prie MOP lauko tranzistoriaus p20 užtūros, ir prie MOP lauko tranzistoriaus p23 užtūros, p23 tokiu būdu gauna signalą a^*, kuris invertuotas magistralės a signalo atžvilgiu. Centrinio valdymo įrenginio signalas mode.a patenka į MOP lauko tranzistoriaus p21 užtūrą, o invertuotas variantas $mode.a^*$ iš centrinio valdymo įrenginio patenka į MOP lauko tranzistoriaus p22 užtūrą. Sujungimo tinklas F1 vykdo funkciją $a^*mode.a$ prieš $a^{**}mode.a^*$.

Dviejų p-kanalo MOP lauko tranzistorių p24 ir p25 ištakas/santakas grandinės yra prijungtos tarp maitinimo įtampos +Vcc ir linijos b. Magistralė a yra prijungta prie MOP lauko tranzistoriaus p24 užtūros ir signalas ba, ateinantis iš centrinio valdymo įrenginio, patenka į MOP lauko tranzistoriaus p25 užtūrą.

Taškas tarp tinklo F1 ir MOP lauko tranzistoriaus n23 santako yra prijungtas prie n-kanalo MOP lauko tranzistoriaus n24, kurio ištakas prijungtas prie žemės ir santakas prijungtas prie magistralės 12 (MODE), skirtos režimo signalams ir parodytos 2 ir 3 piešiniuose, užtūros. Dviejų p-kanalo MOP lauko tranzistorių p26 ir

p27 ištakas/santakas grandinės yra prijungtos tarp maitinimo įtampos +Vcc ir linijos b. Režimo signalas magistralėje 12 per du invertorius INV 21 ir INV 22, sustiprinančius šį signalą, patenka į MOP lauko tranzistoriaus p27 užtūrą. Signalas mode.b iš centrinio valdymo įrenginio yra paduodamas į MOP lauko tranzistoriaus p26 užtūrą.

Signalas mode.a vykdo operaciją mode magistralėje a. Signalas mode.a* taip pat vykdo šią operaciją, bet magistralėje suformuoja invertuotą reikšmę a*. Signalas ba užstato magistralėje aukštą lygį, kai magistralėje a yra žemas lygis. Signalas mode.b užstato magistralėje b aukštą lygį, kai magistralėje režimas yra 'žemas lygis'. Signalas prech užstato magistralėje a ir b žemą lygį.

MODE operacijos vykdymas signalui a arba a* yra realizuojamas sujungimo tinklo F1 pagalba. Magistralėje MODE 12 per visas išraiškų viršūnes 17 gali būti nustatytas žemės lygis. Tranzistoriaus p26 užtūroje esantis valdymo signalas užstato magistralėje b aukštą lygį. Taip galima įvykdyti du skirtingus testus kaip ir magistralėje a taip ir magistralėje b. Jeigu testas parodo, kad vykdomos operacijos metu magistralėje a yra aukštas lygis, magistralėje b gali būti užstatytas aukštas lygis. Pvz. operacija WIRED OR turėtų būti padaryta vienoje iš magistralių a ir b, ir kita operacija WIRED AND vykdoma kitoje ir operacijos metu surasta, kad pvz. magistralėje a yra reikšmė 'neteisinga', tada magistralėje b turėtų būti reikšmė 'neteisinga'. Kitų operacijų sąlygos remiasi magistralės b reikšme.

Dviejų p-kanalo MOP lauko tranzistorių p28 ir p29 ištakas/santakas grandinės yra prijungtos tarp maitinimo įtampos +Vcc ir linijos b. Prioritetų dekoderio 11 signalas grant per invertorių INV 23 prijungtas prie NE-IR loginio elemento NAND1 įėjimo, prie kito šio elemento įėjimo prijungtas signalas req. NE-IR loginis elementas įjungia du n-kanalo MOP lauko tranzistorius n25 ir n26, kurių ištakas/santakas grandinės sujungtos kartu, ir du p-kanalo MOP lauko tranzistorius p32 ir p34, kurių ištakas/santakas grandinės sujungtos lygiagrečiai ir prijungtos tarp MOP lauko tranzistoriaus n26 ir maitinimo įtampos Vcc. MOP lauko tranzistoriaus n25 ištakas prijungtas prie žemės. Signalas grant* ateina į MOP lauko tranzistorių n26 ir p34 užtūras ir signalas req

ateina į MOP lauko tranzistorių n25 ir p32 užtūras. NE-IR elemento išėjimas prijungtas prie MOP lauko tranzistoriaus p28 užtūros ir prie p-kanalo MOP lauko tranzistoriaus p33, kurio ištakas/santakas grandinė prijungta tarp maitinimo įtampos Vcc ir linijos MORE 13, užtūros. Signalas grant.b iš centrinio valdymo įrenginio ateina į MOP lauko tranzistoriaus p29 užtūrą.

Dviejų p-kanalo MOP lauko tranzistorių p30 ir p31 ištakas/santakas grandinės yra prijungtos tarp maitinimo įtampos +Vcc ir linijos req, einančios į prioritetų dekoderį 11. Signalas prio iš centrinio valdymo įrenginio patenka į MOP lauko tranzistoriaus p30 užtūrą, ir magistralė b yra prijungta prie MOP lauko tranzistoriaus p31 užtūros.

Signalas prio iš centrinio valdymo įrenginio naudojamas, kai reikalinga pasiųsti pareikalavimo signalą į prioritetų dekoderį 11, t. y. pasiųsti aukšto lygio signalą req, jeigu magistralėje b yra žemas lygis. IR-NE elementai p32, p34, n25, n26 atpažįsta jeigu yra aukštas signalo req lygis ir žemas signalo grant lygis, t. y. jeigu prioriteto pareikalavimas yra, bet jis dar neapdorotas. Tokiu atveju IR-NE elementas numetamas ir nustato magistralę 13 (MORE) į aukštą lygį, t. y. parodo kad egzistuoja prioriteto reikalavimas. Signalas grant.b naudojamas magistralės b užstatymui į aukštą lygį, kai signalas req yra 'aukštas', ir signalas grant yra 'žemas'. Po to visų pirma signalas prio ateina iš centrinio valdymo įrenginio ir po to signalas grant.b mažiausiai vienoje atminties ląstelėje nustatys magistralę b į aukštą lygį.

SKAITMENINIS ALU IR REIKŠMĖS ATSPINDĖJIMAS

Išradime numatyta pirmos eilės procesorių aprūpinti tradicinio tipo skaitmeniniu ALU, kuris valdomas valdymo įrenginio 6 pagalba ir yra sujungiamas su bazine ląstele. Skaitmeninio ALU, kuris pritaikytas specialiai naudoti su išradime aprašytu pirmos eilės procesoriumi, įranga bus aprašyta žemiau. Skaitmeninis ALU, aprašytas žemiau:

* gali redukuoti skaitmenines išraiškas nenaudodamas tarpinės (laikinos) atminties
* gali dirbti kaip su plaukiojančio kablelio skaičiais taip su operandais, turinčiais kintantį mantisinės ir eksponentinės

dalies padalinimą, kurio padėtis užkoduota kodo lauke

* naudoja aritmetines instrukcijas pateiktas operanduose, magistralėse, kintančio kodo ilgio lauke.
* realizuoja kovariaciją tarp plaukiojančio kablelio ir sveiko atspindėjimo
* realizuoja kodo tankumo reikšmės atspindėjimą

Skaitmeninis ALU gali vykdyti aritmetines, logines ir santykių operacijas su skaitmeninėmis elementų reikšmėmis sekančiais būdais:

a) apdorojami įėjimo sąrašas su skaitmenininėmis reikšmėmis ir instrukcijos

b) operacija atliekama su įėjimo sąrašu su skaitmenininėmis reikšmėmis naudojant instrukcijų informaciją

c) skaičiavimai atliekami perrašant elementus įėjimo sąraše

d) rezultatas atspindimas išėjimo sąraše.

Maksimalus elementų skaičius įėjimo ir išėjimo sąrašuose sudaro keturis, šis skaičius atitinka bazinės ląstelės pagrindinių registrų, gaunančių įėjimo sąrašą ir pateikiančių išėjimo sąrašą, skaičių. Vienas sąrašo elementas gali būti atspindėtas kaip instrukcija, du iš elementų gali būti vieno iš dviejų skaitmeninių formatų ir vienas elementas naudojamas tarpiniam skaičiavimo rezultatui laikyti, kai įėjimo elementai skaičiavimo metu kelis kartus perrašomi ar kitaip apdorojami.

Sąrašas įjungia funkciją, kurioje vienas elementas yra instrukcijos kodas, o likę yra instrukcijos argumentai. Instrukcija yra vykdoma perrašant ir perorganizuojant instrukcijos kodą iš išįjimo sąrašo į įėjimo sąrašą kol bus gautas galutinis rezultatas. Kiekvieno perrašymo metu perrašytas sąrašas įjungia pertvarkytą instrukcijos kodo žodį, paskui kurį seka reikšmių žodžiai.

Paprastai prieš apdorojimą skaitmeniniai reikšmių žodžiai peteikiami koduotoje formoje. Paprastai atspindintys skaitmenininę reikšmę žoddžiai pateikiami dažnuminiame skaitmeninio žodžio kodavime, t. y. kiekviena užkoduota reikšmė atitinka tik vienai interpretuojamai reikšmei. Pirmasis kodavimo tipas taikomas binariniam žodžiui, atspindinčiam sveiką reikšmę. Antrasis kodavimo tipas taikomas binariniam žodžiui, atspindinčiam plaukiojančio

kablelio reikšmę. Kodavimas yra tokio pobūdžio, kad po kodavimo plaukiojančio kablelio reikšmės pateikiamos ta pačia tvarka, kaip ir sveikos reikšmės. Tokiu būdu nei viena tipas nereikalauja dvejetainio skaičio formos. Aparatūrinė įranga leidžia atlikti skaičiavimus su transformuotais bitų piešiniais pagal tam tikras taisykles, t. y. nekreipiant dėmesio į tai kokios yra padarytos interpretacijos.

Lengviausias kelias pasiekti plaukiojančio kablelio skaičiaus su kintamu eksponentės ilgiu atspindėjimo formą yra tai, kad dvejetainė koduota plaukiojančio kablelio skaičiaus forma įjungia kodo lauką, eksponentės lauką ir mantisės lauką, kur kodo laukas turi žymę, nurodančią dalinimo ribą tarp eksponentės ir mantisės laukų, ir tokiu būdu mantisė ir eksponentė gali turėti skirtingo ilgio laukus. Dažnuminis pateikimas pasiekiamas apibrėžiant virtualų "1", t. y. "1" fiziškai nėra realizuotas, plaukiojančio kablelio formos mantisės dalies lauko priekyje.

Skaitmeninio ALU aparatūrinė įranga

Skaitmeninis aritmetinis loginis įrenginys ALU, parodytas pieš. 24A, įjungia valdymo įrenginį 1a ir aritmetinį įrenginį 2a, atliekantį aritmetines operacijas įėjimuose/išėjimuose NU_0, NU_1, NU_2 ir NU_3, kurių kiekvienas turi daugelį (M) perdavimo linijų, kuriomis perduodama M bitų reikšmė, kur pvz. M yra 32. Skaitmenis ALU yra valdomas valdymo įrenginio 6 pagalba. Skaitmeninis ALU bendrauja su valdymo įrenginiu 6 sekančių signalų pagalba: taktinio signalo įėjimas C_{ALU}, užimtumo išėjimas A_{BUSY}, valdymo įėjimas A_{CNTL}, būsenos išėjimas A_{STATE}, ir dekoduotos instrukcijos išėjimas A_{INS}.

Interfeisas: skaitmeninis ALU<->Bazinė ląstelė

Magistralės įėjimai/išėjimai NU_0, NU_1, NU_2 ir NU_3 yra prijungti prie pirmos eilės redukuojančio procesoriaus (pieš. 1), kuriame vykdomos redukavimo instrukcijos, bazinės ląstelės pagrindinių registrų.

Bazinės ląstelės pagrindinių registrų ląstelės $S_{0,0}$, $S_{1,0}$, $S_{2,0}$, $S_{3,0}$ (pieš. 9) yra prijungtos prie terminalo NU (pieš. 10), kuris yra prijungtas prie vieno iš įėjimų/išėjimų $NU_0 \div NU_3$. Tokiu būdu pagrindinio registro $S_{X,0}$ turinys sąrašo pavidalu gali būti

perduotas per įėjimo/išėjimo magistralę NU_x, kur x yra tarp 0 ir 3. Skaitmeninis ALU grąžina rezultatą išėjimo sąrašo pavidalu į pagrindinius registrus naudodamas tą pačią magistralę. Pagrindinių registrų, veikiančių NUM plokštumoje, t. y. atspindinčių skaitmeninius žodžius, turinys yra perduodamas į skaitmenininį ALU.

Įėjimų/išėjimų magistralės $NU_0 \div NU_3$ perduoda skaitmeninių žodžių, kurie gali būti reikšmės arba instrukcijos, sąrašą. Gali būti naudojamos ne visos magistralės. Įėjimų/išėjimų $NU_0 \div NU_3$ įėjimo sąrašas įjungia informaciją apie skaičiavimo tipą, ir valdymo įrenginys 1a gauna šią informaciją skirtingų masyvo 2a dalių valdymo metu. Skaičiavimai atliekami perrašant įėjimo sąrašo turinio bitus ar jų grupes, ir perduodant rezultatą į išėjimo sąrašą. Reikia pažymėti, kad ši operacija gali kartotis tam tikrą skaičių kartų, kol išėjimo sąraše bus gautas galutinis rezultatas. Kartojimo metu išėjime bus laikinas išėjimo sąrašas, kuris naudojamas kaip laikinas įėjimo sąrašas.

Sąraše gali būti funkcija su savo pradinėmis reikšmėmis, galimas ir kitas sąrašo variantas, kaip sąrašas tik mažiausiai su viena sveika ar plaukiojančio kablelio reikšme. Jeigu sąraše yra funkcija, pirmasis sąrašo elementas, ateinantis įėjimų/išėjimų NU_0 pagalba, įjungia instrukcijos kodą, t. y. informaciją apie būsimo skaičiavimo tipą, o kiti sąrašo elementai yra instrukcijos argumentai.

Instrukcija vykdoma ją perrašant vieną ar kelis kartus, kad gauti galutinį rezultatą. Skaitmeninis ALU naudoja savo mikroinstrukcijas ir būsenos signalus. Juos naudoja skaitmeninio ALU viduje esantis valdymo įrenginys 1a masyvui 2a valdyti. Kai tiesiogiai nurodytas perrašymas padarytas, rezultatas patalpinamas į išėjimo sąrašą. Jei įvykdytas žingsninis perrašymas, perrašytas sąrašas išsaugo tą patį arba šiek tiek modifikuotą instrukcijos kodą, paskui kurį daugeliu atveju seka reikšmė arba reikšmės. Reikia pažymėti, kad perrašymų metu geriausia išsaugoti instrukcijos kodą, tokiu atveju valdymo įrenginys 6 bus mažiau apkrautas.

Interfeisas skaitmeninis ALU<->centrinis valdymo įrenginys

Grįžtant prie pieš. 24A, taktinis signalas C_{ALU} naudojamas taktiniams impulsams į skaitmenininį ALU pateikti, ir skaitmeninis ALU signalo A_{BUSY} pagalba nurodo ar jis pasiruošęs priimti

taktinius impulsus.

Pagrindinių registrų dalių turinys, veikiantis HEAD plokštumose, t. y. požymių žodžiuose, yra aptarnaujamas iš centrinio valdymo įrenginio 6, kuris kartu valdo skaitmeninio ALU valdymo įrenginį 1a.

Išradime numatyta ir požymių įjungimas į informaciją, pateikiamą per įėjimus/išėjimus, ir tokiu būdu požymių informacija pateikiama tiesiai valdymo įrenginiui, žemiau dar vadinamu skaitmeninio ALU valdymo grandine.

Valdymo įėjimas A_{CNTL} naudojamas skaitmeninio ALU skaitmeninėms operacijoms valdyti. Šios operacijos gali trukti keletą taktinių ciklų.

Išėjimas A_{STATE} grąžinamas į centrinio valdymo įrenginį 6 po kiekvienos operacijos. Išėjime A_{INS} esantis instrukcijos kodas yra pagrindinio registro, prie prijungta magistralė NU_0, instrukcijos kodo suspausta forma.

Pagal pieš. 24B, bazinis dažnis C_{10}, teikiamas procesoriui iš centrinio valdymo įrenginio 6, gali būti pvz. su periodu 10 ns. Skaitmeninio ALU operacija atliekama per vieną ar kelis tokius taktinius ciklus. Prijungto prie skaitmenininio ALU bazinio procesoriaus periodas yra ilgesnis, pvz. du kartus, negu skaitmeninio ALU periodas, t. y. jegu C_{10} yra 10 ns, tai C_{20} bus 20ns. Galimas ir variantas, kai šis periodas yra trumpesnis. Bazinis valdymo signalas C_{ALU} su trumpesniu periodu, instrukcijos vykdymo gale yra užlaikomas tam kad sinchronizuotis su taktiniu signalu C_{20}, kurio periodas yra ilgesnis. Skaitmeninis ALU generuoja aukštą signalą A_{BUSY}, jeigu jam reikia daugiau C_{10} ciklų instrukcijai įvykdyti. Užlaikymas padaromas po to, kai signalas A_{BUSY} tampa žemas (žiūr. pieš. 24B).

REIKŠMĖS ATSPINDĖJIMAS

Skaitmeniniame ALU plaukiojančio kablelio skaičiai turi kintančią ribą tarp eksponentės ir mantisės. Tokiu būdu, visų pirma galima panaudoti pakankamai daug mantisės bitų mažiems skaičiams atspindėti, kas leidžia pateikti juos su dideliu tikslumu. Iš kitos pusės gali būti realizuoti labai dideli skaičiai.

Kaip parodyta pieš. 25, kiekvienas magistralės bitinis piešinys,

atitinkantis tam tikrą reikšmę, savo reikšmingiausiuose bituose sudaro trumpą bitinį lauką, kuriame yra informacija apie skaičiaus ženklą, eksponentės ženklą ir kodas, nurodantis ribą tarp skaitmeninės reikšmės mantisės m ir eksponentės e (pieš. 25). Reikia pažymėti, kad kad kodinės dalies ilgis gali kisti, ir riba tarp kodo lauko ir eksponentės lauko yra apibrėžiama kodo lauke (pieš. 25).

Skaitmeninis ALU skaičiaus plaukiojančio kablelio formą atspindi taip pat, kaip ir to pačio skaičiaus sveikos formos interpretaciją. Tokiu būdu plaukiojančio kablelio reikšmė ir sveika reikšmė, turinčios tą patį bitinį piešinį, yra skirtingos.

Papildant galima pasakyti, kad plaukiojančio kablelio formos bitinėms eilutėms interpretuoti naudojama vienas mechanizmas, o sveikos formos interpretacijai - kitas mechanizmas. Tokiu būdu tampa nereikalinga dvejetainė forma, vietoj to mažiausiai vienas iš dumenų tipų (sveikas ar plaukiojančio kablelio), geriausiai abu, yra koduotoje formoje. Bitinio piešinio transformacija daroma automatiškai, ir visi skaičiavimai atliekami su transformuotomis reikšmėmis nežinant apie padarytas interpretacijas.

Svarbiausia yra, kad plaukiojančio kablelio reikšmės turi būti atspindimos vienareikšmiškai, t. y. negali būti dviejų bitinių piešinių, kurie atspindėtų tą patį skaičių.

Grafinis sveiko skaičiaus atspindėjimas

Pieš. 26A parodyta išradime aprašyto sveikų skaičių kodo, vadinamo sveikomis H reikšmėmis, grafinio pateikimo forma. Dvejetainis atspindėjimas pateiktas horizontalioje ašyje, o H reikšmės - vertikalioje ašyje. Reikšmingiausias bitas atspindi reikšmės ženklą - teigiamą ar neigiamą. Kaip matyti iš pieš, 26A H reikšmė turi didžiausią neigiamą reikšmę, kai visi bitai yra "0", nulinę reikšmę, kai visi bitai, išskyrus reikšmingiausią, kuris yra "1", yra "0", ir didžiausią teigiamą reikšmę, kai visi bitai yra "1"- t. y. maksimalią reikšmę max_{rep}.

Grafinis plaukiojančio kablelio skaičiaus atspindėjimas

Pieš. 26B parodyta išradime aprašyto plaukiojančio kablelio skaičių kodo, vadinamo plaukiojančio kablelio H reikšmėmis, grafinio pateikimo forma. Dvejetainis atspindėjimas pateiktas

horizontalioje ašyje, o H reikšmės - vertikalioje ašyje, kurioje ašių kirtimo taškas atspindi plaukiojančio kablelio H reikšmių ženklo virtimą. Reikšmingiausias bitas atspindi reikšmės ženklą. Plaukiojančio kablelio reikšmės atspindėjimas turi patenkinti universalumo reikalavimą. Reikalingam tikslumui posiekti reikšmės dydis turi kisti. Tokiu būdu galimas labai didelių ir labai mažų reikšmių atspindėjimas. Plaukiojančio kablelio H reikšmių skalė yra logaritminė. Logaritminės H reikšmės gali būti paprastai tiesiškai aproksimuojamos 2 daugikliais. Rezultate gaunamas tipinis plaukiojančio kablelio kodas. Kaip matyti iš pieš. 26B dvejetainis H reikšmių tarp -1 ir +1 atspindėjimas gaunamas naudojant dvejetainio lauko vidurinę dalį, kuri yra pusės šio lauko dydžio.

Santykinis tikslumas yra tam tikrose reikšmių diapazone yra pastovus. Logaritminės reikšmės turi tam tikrą tikslumą ir diapazoną. Jeigu reikšmės dydis yra pakankamai didelis, gali būti pakeičiamas diapazonas.

Logaritminėms reikšmėms gali būti kelis kartus panaudota logaritminė funkcija, ir tokiu būdu yra įmanoma dirbti su labai didelėmis reikšmėmis. Tačiau esant didelėms reikšmėms, krenta tikslumo lygis, nors mažoms reikšmėms jis yra išlaikomas ar net truputį padidėja.

Kadangi dviejų bitinių piešinių atspindėjimai gaunami identišku būdu, sveikų skaičių dydžio sulygintojas (komparatorius) tinka ir plaukiojančio kablelio skaičių atspindėjimams sulyginti. Tai reiškia, kad nerikalingas specialus plaukiojančio kablelio skaičių komparatorius. Tai reiškia, kad sveikų skaičių dydžio sulygintojas (komparatorius), kuris atlieka sulyginimo operaciją per vieną ciklą, gali būti naudojamas plaukiojančio kablelio reikšmėms sulyginti. Tradicinio tipo plaukiojančio kablelio reikšmių komparatorius sulyginimo operacija trunka daugelį ciklų.

Tradiciniuose aritmetiniuose loginiuose įrenginiuose keletas bitinių piešinių gali būti interpretuoti kaip tas pats skaičius, ir todėl normalizacijos pagalba išsirenkamas vienas iš galimų variantų. Išradime aprašytame įrenginyje normalizacija tampa nereikalinga (bent jau vykdomoje operacijoje). Normalizacija gali būti vykdoma prieš ir po skaičiavimo operacijų. Tiksliai atspindint plaukiojančio kablelio reikšmių panašumą, plaukiojančio kablelio

skaičiavimai, tokie kaip sulyginimas, gali būti atlikti tik per vieną ciklą. Tikslus atspindėjimas taip pat leidžia panaudoti maksimalų plaukiojančio kablelio reikšmių skaičių.

Vienas iš atvejų, kai tradiciniuose uždaviniuose keletas piešinių gali būti interpretuoti kaip tas pats skaičius yra tai, kad reikšmės eksponentinė dalis ir reikšmės mantisės dalis yra dauginamos tarpusavyje tam, kad gauti pačią reikšmę. Pvz. reikšmė 1.0 gali būti $2^1*1/2$, t. y. eksponentės bitinis piešinys yra 01 ir mantisės bitinis piešinys yra 100, arba reikšmė gali būti $2^2*1/4$, t. y. eksponentės bitinis piešinys yra 10 ir mantisės bitinis piešinys yra 010 (eksponentės bitinis piešinys ir mantisės bitinis piešinys eina vienas paskui kitą).

Tikslus atspindėjimas realizuojamas kai naudojamas sąlyginis '1' prieš mantisės bitinį piešinį teigiamoms reikšmėms, ir sąlyginis '0' neigiamoms reikšmėms. Tradiciškai mantisė yra apibrėžiama reikšmėmis tarp -1 ir +1. Dvejetainės mantisės diapazonas yra 1.0÷2.0 arba -2.0÷-1.0 - proklausomai nuo ženklo.

Atspindėjimo kodavimas

Sveikų skaičių piešiniai yra tiksliai nustatyti, plaukiojančio kablelio skaičiams nustatyti tą pačią atspindėjimo tvarką, kaip sveikiems, yra pakankamai sunku.

Šios problemos sprendimas yra parodytas pieš. 26C. Pieš. 27A - 27D parodytas duomenų žodžių sąrašas sutvarkytas pagal principus, parodytus pieš. 26C. Pieš. 26C parodyta grafinė plaukiojančio kablelio kodo forma, dvejetainis variantas parodytas į kairę nuo linijos LIN, plaukiojančio kablelio H reikšmės parodytos linijos LIN dešinėje. Linija LIN yra ašis, nukreipta didėjančių reikšmių (tiek dvejetainių, tiek H-reikšmių) link. Dvejetainės ir H-reikšmės keičiasi einant nuo linijos LIN. Artimiausias prie LIN grafikas atitinka pieš. 26B ašis. Kairėje yra H-reikšmių ašis, o dešinėje yra dvejetainė ašis. Sekantys grafikai, tolstant nuo LIN, sh dešinėje ir s kairėje atitinka reikšmingiausią bitą. Jis yra '0' neigiamoms reikšmėms ir '1' teigiamoms reikšmėms. Sekančio lauko grafikas, atitinkamai seh ir se, atitinka sekantį reikšmingiausią bitą. Tai yra eksponentės ženklo bitas. Sekantis laukas įjungia mažiausiai vieną bitą, kurio reikšmingumas yra mažesnis negu sekančio reikšmingiausio bito, vienas bitas yra kai dvejetainis

laukas yra mažas, kas parodyta septynių bitų atveju, ir daugiau kaip vienas bitas yra kai dvejetainis laukas yra didelis, kas vėliau bus iliustruota 32 bitų atveju.

Pavyzdyje, parodytame pieš. 26C, 27A - 27D naudojamas septynių bitų žodis (vietoje 32 bitų), kas be abejo praktikoje dažniau sutinkama. Dvejetainis pateikimas kairėje nuo LIN rodo H-reikšmių kitimą nuo -1024 iki +768. Reikia pažymėti, kad H-reikšmės -1 ir +1 yra pusiaukelėje tarp 0 ir atitinkamų didžiausių reikšmių -1024 ir 768. Pirmasis bitas s ir sh tiek dvejetainėje, tiek H-reikšmių formose atspindi reikšmės ženklą - teigiamą ar neigiamą.

Sekantis dvejetainis bitas yra eksponentės ženklas, t. y. parodo ar reikšmė yra didesnė ar mažesnė už 1. H-reikšmės eksponentės ženklas yra teigiamas tarp -1024 ir -1, neigiamas tarp -1 ir +1, toliau vėl teigiamas tarp +1 ir +768. Kaip matyti iš pieš. 26C kairės pusės, dvejetainės formos eksponentės ženklas se yra neigiamas pirmame ketvirtadalyje, teigiamas antrame, neigiamas trečiame, ir teigiamas ketvirtame. Bitas seh gali būti išreikštas kaip jegu s=1 tai seh=se, kitaip seh=1-se.

Taip atspindimas reikšmingiausias bitas, pažymintis simbolio ženklą, kinta lėčiausiai, sekantis pagal reikšmingumą bitas, pažymintis eksponentės ženklą, kinta greičiau, t. y. dvgubai greičiau, negu simbolio ženklo bitas. Sekantis bitas kinta dvigubai greičiau, negu eksponentės bitas ir yra kodo bitas c. Kodo laukas pavyzdžiuose pieš. 26C ir 27A - 27D susideda tik iš vieno bito. Kituose pavyzdžiuose kodo laukas gali susidėti ir iš daugiau bitų. Kodas kontroliuoja eksponentės ilgį ir atsižvelgia į ženklų bitus.

Plaukiojančio kablelio reikšmė, kuri atspindi uždavinį, t. y. plaukiojančio kablelio pažymėjimas, tvarko kodo lauką ch specialiu būdu. Kaip matyti iš sekcijos eh, kurioje tiesiogiai pateikta H-reikšmės eksponentė, reikšmė eh keičiasi nuo didžiausios teigiamos reikšmės iki 0, kur H-reikšmė yra -1, toliau keičiasi iki didžiausios neigiamos reikšmės, kur H-reikšmė yra 0, nuo čia iki reikšmės 0, kur H-reikšmė yra +1, ir toliau iki didžiausios teigiamos reikšmės, kur H-reikšmė yra 768. Sekcijoje eh_{abs} parodytos absoliutinės eksponentės H-reikšmės kaip dvi kreivės, einančios viena šalia kitos, ir kurių galuose yra didžiausios teigiamos reikšmės, o viduryje - 0. H-reikšmių kodo laukas yra

padarytas atspindėti H-reikšmes eh_{abs} ir bus parodytos piešinyje brūkšnine linija formos ir parodytos piešinyje ištisa linija formos, jeigu kodo laukas yra tik vieno bito dydžio. Taip ch=c jeigu se=1, ir kitu atveju ch=1-c, kai kodas ch turi tik vieną bitą.

Yra labai lengva suprasti pieš. 27A - 27D parodyto sąrašo struktūrą, kadangi kiekvienas bitinis laukas s, sh, se, seh, c ir ch yra tik vieno bito dydžio, ir pateikimų pieš. 26 ir pieš. 27A - 27D sulyginimas yra paprastas. Pirmoje pieš. 27A - 27D pateiktame stulpelyje parodyta nuosekli dvejetainių skaičių seka nuo 0000000 iki 1111111, o sekančiame stulpelyje - tos pačios reikšmės dešimtainėje sistemoje. Rodyklių stulpelio kairėje parodytos transformuoti skaičiai ir H-reikšmės. Sąrašas padalintas ištisinėmis horizontaliomis linijomis lengvesniam sulyginimui su pieš. 26C.

ch stulpelio dešinėje esantis stulpelis įjungia formos eksponentės reikšmės e dešimtainį variantą ir šalia yra H-reikšmės eksponentės reikšmė dešimtainiame pavidale. Sekančiuose dviejuose stulpeliuose parodytos e ir eh dvejetainės reikšmės. Dar dviejuose sekančiuose stulpeliuose parodyto mantisės m reikšmės dešimtainiame ir dvejetainiame pavidale. Po to parodyta H-reikšmės mantisės reikšmė mh. Reikia pažymėti, kad mantisė kinta nuo -2 iki -1 pieš. 27A ir 27B, kur reikšmė sh yra "0", ir +1 iki +2 pieš. 27C ir 27D, kur reikšmė sh yra "1". Dešiniausiame stulpelyje parodyta dešimtainė H-reikšmė, atitinkanti pirmojo stulpelio reikšmę po transformacijos.

Kaip matyti iš apatinės pieš. 27C dalies, kai atspindimos formos reikšmingiausias bitas yra "1", o kiti - "0", tai atspindima reikšmė yra plaukiojančio kablelio reikšmė "0".

Kaip matyti iš dvejetainių skaičių pieš. 26C, parodytų linijos LIN kairėje, iš septynių bitų žemiausieji keturi yra skirti mantisei ir eksponentei, kadangi trys aukščiausi bitai rezervuoti ženklams ir kodui. Iš skaitmenų stulpelių e ir m matyti, kad eksponentės ženklas yra negiamas, t. y. "0", ir kodo c reikšmė yra "1", reikšmė e turi tris bitus, o m atitinkamai vieną (pirmieji du nuliai stulpelyje turi būti praleisti). Tai yra parodyta pieš. 26C. Sekančiu atveju, kai eksponentės ženklas se yra neigiamas, bet kodo

c reikšmė yra "0", reikšmė e turi tik vieną bitą (pirmieji du nuliai stulpelyje turi būti praleisti), o mantisės reikšmė turi tris bitus (taip pat parodyta ir pieš. 26C). Toliau sekančioje dalyje reikšmė e įjungia tik vieną bitą, o mantisė - tris ir t. t.

Kaip parodyta, eksponentė e daugelyje kitimo etapų turi reikšmes c=0 ir se=0 ir keliuose c=1 ir se=1. Dalyje e+bias, esančioje į kairę nuo dalies e, tai parodyta ištisinėmis linijomis. Brūkšninėmis linijomis parodyta kaip reikšmės eh_{abs} padaromos invertuojant kas antrą ketvirtadalį e+bias dalyje (eh yra e reikšmė su eksponentės ženklu).

Tokiu būdu kodo bitas ar bitai priklauso nuo ženklų bitų, ir todėl galima sakyti kad faktiškai kodas įjungia ženklų bitus ir turi būti riboto dydžio.

Mantisės ilgis priklauso nuo eksponentės ilgio, kadangi jeigu eksponentė yra ilga, tai mantisė turi būti trumpa ir atvirkščiai. Todėl mantisė nera parodyta pieš. 26C, bet pateikta pieš. 27A -27D m,m stulpelyje, kur pirmasis m yra mantisės dešimtainė išraiška, o antrasis m - dvejaetainė. Kai s=1, tada mh=1.0+m, kitu atveju mh=-2+m, kaip pažymėta aukščiau.

Pateiktas pavyzdys skirtas aiškumo dėlei ir todėl liečia setynių bitų atvejį. Kaip buvo minėta, žodžio ilgis gali būti didelis, pvz. siekti 32 bitus. tada yra prasmė turėti kintančio ilgio kodo lauką. Taip pat yra geriau, kai riba tarp mantisės ir eksponentės kečiama tam tikro dydžio žingsniais, pvz. per keturis bitus. Privalumas yra tame, kad nereikia didelio kodų skaičiaus sudalinimui tarp mantisės ir eksponentės reguliuoti. Tokio tikslumo dekoderio pavyzdys bus aprašytas žemiau.

SKAITMENINIO ALU ĮRANGA

Pieš. 28 parodyta smulki skaitmeninio ALU įranga, kuris pagal išradimą gali būti įjungtas į pirmos eilės redukuojantį procesorių. Įėjimų/išėjimų magistralės $NU_0 \div NU_3$ prijungtos prie įėjimų/išėjimų buferio 20, kuris perduoda įėjimo informaciją į vidines linijas $v_{0i} \div v_{3i}$ taktinio dažnio, ateinančio į buferio įėjimą signalo CLOCK pavidalu iš išorinio generatoriaus, kiekvieno invertuoto takto metu, ir kuris perduoda vidinių išėjimo linijų $v_{0out} \div v_{3out}$ reikšmes į įėjimų/išėjimų magistrales $NU_0 \div NU_3$ kiekvieno takto metu.

Grandinės 20 schema ir jos taktavimas parodyta pieš. 29A, 29B ir 29C. Įėjimo/išėjimo buferis 20 įjungia įėjimo buferius, po vieną kiekvienai įėjimo/išėjimo magistralei $NU_0 \div NU_3$, ir tokį pat skaičių išėjimo buferių. Taktiniai signalai neturi persidengti, kas pasiekiama nukerpant priekinius frontus.

Pirmosios CLOCK signalo fazės metu (pieš. 29A) kai signalas yra invertuotas bazinės ląstelės pagrindinių registrų reikšmės per įėjimą ir valdomus įėjimo buferius patenka į vidines skaitmeninio ALU duomenų linijas. Duomenys toliau sklinda šiomis linijomis ir perduodami į sekančios būsenos metu į išėjimo magistrales. Išėjimo buferiai laikomi užblokuoti.

Kaip parodyta pieš. 29B, taktinio signalo CLOCK periodo 1 fazės metu, pažymėto skliausteliuose kairėje, įėjimo buferiai grandinėje 20 užblokuojami ir vidinės skaitmeninio ALU grandinės dirba kaip multipleksoriai ir kontroliuoja signalus siunčiamus į įėjimų/išėjimų magistrales $NU_0 \div NU_3$. Išėjimo buferiai grandinėje 20 pervedami į siuntimo būseną.

Daugiaciklės instrukcijos sąlygos 0 fazės metu yra parodyta pieš. 29C. Įėjimo/išėjimo magistralių $NU_0 \div NU_3$ turinys, gautas iš skaitmeninio ALU prieš tai buvusios fazės 1 metu ir dėl talpumo laikomas jose visą 0 fazę, patenka per įėjimo buferius, kurie yra atidaryti, kai tuo tarpu išėjimo buferiai yra užblokuoti, į įėjimo magistrales v_{0i}, v_{1i}, v_{2i}, v_{3i}, ir toliau sklinda skaitmeninio ALU vidinėmis duomenų linijomis, kol šis nepereina į sekančią būseną. Įėjimo sąrašas naudojamas skaičiavimų valdymui ir vykdymui, kurie realizuojami perrašanat įėjimo sąrašo turinį ir prduodant jį į išėjimą. Pirmasis sąrašo elementas yra instrukcijos kodas, o likę elementai yra instrukcijos, reikalaujančios tam tikrų aparatūrinės dalies veiksmų, argumentai. Tam tikra informacija priklausomai nuo magistralių v_{0i}, v_{1i}, v_{2i}, v_{3i} reikšmių yra gaunama įėjime EXT iš centrinio valdymo įrenginio 6.

Centrinio valdymo įrenginys 6 gauna informaciją ar žodžiai magistralėse v_{0i}, v_{1i}, v_{2i}, v_{3i} atspindi instrukcijas ar/ir skaičius, ar tie skaičiai yra sveiki ar plaukiojančio kablelio ir perduoda šią informaciją į skaitmeninio ALU valdymo grandinę 27 per magistralę EXT.

Sekantys skaičiavimo įrenginiai naudojami pieš. 28 pavaizduotoje

schemoje skirtingoms operacijoms atlikti:

Komparatoriai (sulyginimo įtaisai)

Pirmasis komparatorius 21, prijungtas prie linijų v_{0i} ir v_{1i}, pilnai sulygina šių linijų informaciją, t. y. $v_{0i}>v_{1i}$, $v_{0i}=v_{1i}$ ir pateikia sulyginimo rezultatus, t. y. dviejų bitų reikšmę, į savo išėjimą cmp_{01}, prijungtą prie skaitmeninio ALU valdymo grandinės 27. Antrasis komparatorius 22, prijungtas prie linijų v_{1i} ir v_{2i}, pilnai sulygina šių linijų informaciją, t. y. $v_{1i}>v_{2i}$, $v_{1i}=v_{2i}$ ir pateikia sulyginimo rezultatus, t. y. dviejų bitų reikšmę, į savo išėjimą cmp_{12}, prijungtą prie skaitmeninio ALU valdymo grandinės 27. Trečiasis komparatorius 221, prijungtas prie linijų v_{2i} ir v_{3i}, pilnai sulygina šių linijų informaciją, t. y. $v_{2i}>v_{3i}$, $v_{2i}=v_{3i}$ ir pateikia sulyginimo rezultatus, t. y. dviejų bitų reikšmę, į savo išėjimą cmp_{23}, prijungtą prie skaitmeninio ALU valdymo grandinės 27. Ketvirtasis eksponentės komparatorius 222 yra prijungtas prie linijų v_{1i} ir v_{2i}, kurių kiekviena yra 32 bitų magistralė, ir sulygina reikšmingiausias šių linijų žodžių dalis, t. y. dalis se, c, e, ir t. t. Komparatorius pažymi dalių se, c,e kodavimo tipą. Signalas B_{1sb1}, ateinantis iš tikslumo dekoderio PD, patenka į atskirą komparatoriaus 222 įėjimą.

Penktasis komparatorius 223 sulygina gautus iš tikslumo dekoderio signalus B_{1sb1} ir B_{1sb21}. Rezultato signalas cmp_prec rodo, kad signalai B_{1sb1} ir B_{1sb21} yra identiški.

Valdymo grandinė 27 komparatorių sulyginimo rezultatus perduoda į išėjimą A_{STATE}, prijungtą prie centrinio valdymo įrenginio 6 (pieš. 24A). Rezultatas taip pat gali būti panaudojamas ir kitų įrenginių valdymui, kaip aprašyta žemiau, taip kad išrinkti žodžiai perduodami į išėjimo magistrales v_{1out}, v_{2out} ir v_{3out}.

Nereikšmingumo dekoderis

Nereikšmingumo dekoderis 23 yra prijungtas prie linijos v_{1i} ir tikrina ar visi magistralės v_{1i} bitai yra "1" ar"0", jeigu tai sveikas skaičius, ir, jeigu magistralėje v_{1i} yra plaukiojančio kablelio skaičius, atskirai tikrina eksponentės ir mantisės bitus - ar visi bitai yra "1" ar "0". Signalas B_{1sb1}, indikuojantis paskutinę mažiausio reikšmingumo bitų grupę magistralėje v_{1i}, nurodo ar žodis įėjimo magistralėje yra sveiko ar plaukiojančio

kablelio tipo, ir jeigu tai plaukiojančio kablelio reikšmė, tai nurodoma ribos tarp eksponentės ir mantisės padėtis. Šių testų rezultatai pateikiami išėjime $insig_1$, prijungto prie valdymo grandinės 27. Nereikšmingumo dekoderis 24 yra prijungtas prie linijos v_{2i} ir tikrina ar visi magistralės v_{2i} bitai yra "1" ar"0", jeigu tai sveikas skaičius, ir, jeigu magistralėje v_{2i} yra plaukiojančio kablelio skaičius, atskirai tikrina eksponentės ir mantisės bitus - ar visi bitai yra "1" ar "0". Signalas B_{1sb2}, indikuojantis paskutinę mažiausio reikšmingumo bitų grupę magistralėje v_{2i}, nurodo ar žodis įėjimo magistralėje yra sveiko ar plaukiojančio kablelio tipo, ir jeigu tai plaukiojančio kablelio reikšmė, tai nurodoma ribos tarp eksponentės ir mantisės padėtis. Šių testų rezultatai pateikiami išėjime $insig_2$, prijungto prie valdymo grandinės 27. Nereikšmingumo dekoderis 25 yra prijungtas prie linijos v_{3i} ir tikrina ar visi magistralės v_{3i} bitai yra "1" ar"0", jeigu tai sveikas skaičius, ir, jeigu magistralėje v_{3i} yra plaukiojančio kablelio skaičius, atskirai tikrina eksponentės ir mantisės bitus - ar visi bitai yra "1" ar "0". Signalas B_{1sb1}, indikuojantis paskutinę mažiausio reikšmingumo bitų grupę magistralėje v_{3i}, nurodo ar žodis įėjimo magistralėje yra sveiko ar plaukiojančio kablelio tipo, ir jeigu tai plaukiojančio kablelio reikšmė, tai nurodoma ribos tarp eksponentės ir mantisės padėtis. Šių testų rezultatai pateikiami išėjime $insig_3$, prijungto prie valdymo grandinės 27. Kiekvienas nereikšmingumo dekoderis taip pat valdomas valdymo grandinės 27 signalo $c5_3$ pagalba (bus aprašyta žemiau).

Nereikšmingumo dekoderių pagalba nustatoma ar dalyvaujnti skaičiavimuose informacija yra reikšminga. Gautas rezultatas taip pat pateikiamas į magistralę A_{STATE}, einančią į centrinio valdymo įrenginį 6. Jeigu bent vienas nereikšmingumo įrenginys indikuoja pvz. visus nulius tai gali būti pnaudota valdymo grandinėje 27 tam tikrų tipų operacijų vykdyme.

Instrukcijų dekoderis

Instrukcijų dekoderis 26 yra prijungtas prie linijos v_{0i}. Magistralė v_{0i} gauna instrukciją, kai skaitmeninis ALU vykdo aritmetinę operaciją. v_{0i} dekodavimo resultatas per išėjimą ins siunčiamas į valdymo grandinę 27, jeigu v_{0i} yra instrukcija ar

duomenų reikšmė. Įrenginiai 23÷26 yra taip pat valdomi signalais B_{1sb1}, B_{1sb2} ir c_5, kurie atitinkamai generuojami grandinėse PD1, PD2 ir 27, ir bus aprašyti žemiau.

Sumatoriai ir atėmėjas

Pirmasis sumatorius 28 yra prijungtas prie linijų v_{1i} ir v_{2i}, vykdo sumavimą $v_{1i}+v_{2i}$ ir perduoda rezultatą į atskirą išėjimą a_1. Galimas signalo perdavimas į išėjimą gr_{a1}, prijungtą prie valdymo grandinės 27. Atėmėjas 29 yra prijungtas prie linijų v_{1i} ir v_{2i}, vykdo atėmimą $v_{1i}-v_{2i}$ ir perduoda rezultatą į atskirą išėjimą a_2. Galimas signalo perdavimas į išėjimą gr_{a2}, prijungtą prie valdymo grandinės 27. Antrasis sumatorius 30 yra prijungtas prie linijų v_{1i} ir v_{2i}, vykdo sumavimą $v_{1i}+2*v_{2i}$ ir perduoda rezultatą į atskirą išėjimą a_3. Galimas signalo perdavimas į išėjimą gr_{a3}, prijungtą prie valdymo grandinės 27. Kiekvienas sumatorius yra valdomas valdymo grandinės 27 signalo c5 pagalba ir signalu B_{1ab1}. Šis signalas nurodo ar žodžiai įėjimo magistralėse yra sveiki skaičiai ar plaukiojančio kablelio reikšmės, pastaruoju atveju ir sulygiuotoms magistralių v_{1i} ir v_{2i} reikšmėms nurodo padalinimo ribą tarp eksponentinės ir mantisės dalių. Ši informacija yra svarbi, kadangi diskretiniai dydžiai, t. y. sveiki skaičiai yra vertinami ištisai, o plaukiojančio kablelio skaičiuose atskirai vertinama eksponentė ir atskirai mantisė.

Ženklo grandinės

Pirmoji ženklo ir eksponentės ženklo grandinė 35 išrūšiuoja magistralėje v_{1i} esančio žodžio ženklą, eksponentės ženklą, reikšmingiausią mantisės bitą ir du mažiausiai reikšmingus mantisės bitus ir pateikia savo išėjime $sign_1$ rezultatą, kuris patenka į valdymo grandinės 27 įėjimą. Antroji ženklo ir eksponentės ženklo grandinė 36 išrūšiuoja magistralėje v_{2i} esančio žodžio ženklą, eksponentės ženklą, reikšmingiausią mantisės bitą ir du mažiausiai reikšmingus mantisės bitus ir pateikia savo išėjime $sign_2$ rezultatą, kuris patenka į valdymo grandinės 27 įėjimą. Trečioji ženklo ir eksponentės ženklo grandinė 37 gali būti prijungta prie magistralės v_{3i} ir pateikia savo išėjime $sign_3$ rezultatą, kuris patenka į valdymo grandinės 27 įėjimą. Reikia pažymėti, kad grandinės 36 ir 37 yra neabejotinai reikalingos, kai magistralėse v_{1i} ir v_{2i} esantis žodis turi ženklą, eksponentės ženklą ir

mantisės bitus. Šios grandinės naudojamos skaitmeninio ALU valdymo grandinėje, kai įėjimo magistralėse yra plaukiojančio kablelio reikšmių žodžiai. Grandinė 37 kai kuriose skaitmeninio ALU operacijose gali būti nenaudojama. Ženklo ir eksponentės ženklo grandinė (neparodyta) tam tikrais atvejais realizuota ir magistralei v_{0i}.

Skaitmeninio ALU valdymo grandinė

Valdymo grandinė 27, apdorojusi įėjimo signalus, išleidžia skaitmeninius išėjimo signalus į išėjimo magistrales c1, c2, c3, c4, c5. Valdymo grandinė 27 gali būti loginių elementų staruktūra. Kontrolinių signalų panaudojimo pavyzdžiai bus aprašyti žemiau.

Operandų įrenginys

Du operandų įrenginiai, po vieną kiekvienam iš linijose v_{1i} ir v_{2i} esančių operandų, turi visas aritmetines ypatybes, būdingas šiam elementui. Kiekvienas operandų įrenginys turi kelis informacinius įėjimus. Trys iš jų yra prijungti prie prie atskirų išėjimų a_1, a_2 ir a_3 kurie yra atitinkamuose operandų įrenginiuose 28, 29 ir 30, iš kurių dviejų įėjimai prijungti prie atitinkamų magistralių v_{1i} ir v_{2i}, ir trijų iš jų yra prijungti prie individualaus tikslumo dekoderio PD1 arba PD2, kas bus aprašyta žemiau. Principe kiekvienas informacinis įėjimas yra magistralė, turinti tiek laidų, kiek yra kiekvienoje įėjimo magistralėje v_{0i}, v_{1i}, v_{2i}, v_{3i}.

Operandų įrenginio r1 išėjimas yra išėjimo magistralė v_{1out} ir operandų įrenginio r2 išėjimas yra išėjimo magistralė v_{2out}.

Vienas iš valdymo grandinės 27 išėjimų c1 yra sudėtinis valdymo signalas, skirtas operandų įrenginio r1 vidiniams elementams valdyti. Kiti valdymo signalai operandų įrenginyje r1 yra gaunami iš tikslumo dekoderio PD1. Kitas valdymo grandinės 27 išėjimo sudėtinis valdymo signalas c2 skirtas operandų įrenginio r2 vidiniams elementams valdyti. Kiti valdymo signalai operandų įrenginyje r2 yra gaunami iš tikslumo dekoderio PD2. Trečias valdymo grandinės 27 išėjimo sudėtinis valdymo signalas c3 skirtas operandų įrenginiui r3, ir ketvirtas valdymo signalas c4 patenka į polinomų perdavimo grandinę 31, kuri bus aprašyta žemiau. Penktasis išėjimas c5 valdo tikslumo dekoderius PD1 ir PD2, nereikšmingumo

grandines 23÷25, sumatorius 28, 30 ir atėmėją 29, ir taip pat ženklo ir eksponentės ženklo grandines 35÷37.

Operandų įrenginiai r1 ir r2 turi tokią pačią konfigūraciją, todėl bus aprašytas tik vienas iš jų - r2. Operandų įrenginys yra padalintas į dvi dalis - į eksponentinę dalį OP_e ir mantisės dalį OP_m. Mantisės dalis OP_m turi išėjimą n1 ir eksponentinė dalis turi išėjimą n2. Išėjimai n1 ir n2 yra prijungti prie išėjimo magistralės selektoriaus, vadinamo bitų grupių individualiu selektoriumi M1, įėjimų. 0/1 generatorius 40, išduodantis pasirenkamą tikslumo grupę iš keturių '0' ar '1', valdomas signalo $c2_{11}$ pagalba, ir yra prijungtas prie selektoriaus M1 įėjimo.

Eksponentinės dalies įėjimai yra keturi operandų įrenginio įėjimai, t. y. v_{2i}, B_{bias2}, B_{co_incr2}, B_{co_decr2} iš tikslumo dekoderio PD2, ir įėjimas n8 į kurį ateina signalas iš mantisės dalies OP_m. Įėjimas V_{2i} yra prijungtas prie inkrementoriaus INC įėjimo, prie dekrementoriaus DEc įėjimo ir prie vieno iš penkių žodžių selektoriaus M2 įėjimų. INC išėjimas n5 ir DEC išėjimas n6 yra prijungti prie dviejų žodžių selektoriaus M2 įėjimų. B_{co_incr2}, B_{co_decr2} yra prijungti prie kitų dviejų žodžių selektoriaus M2 įėjimų.

Operandų įrenginio r2 dalis INC tvarko išėjimus gr_{i2} ir Ce_{i2}, o dalis DEC - išėjimus gr_{d2} ir Ce_{d2}. Šie išėjimai yra prijungti prie valdymo grandinės 27. Atitinkamai operandų įrenginio r1 dalis INC tvarko išėjimus gr_{i1} ir Ce_{i1}, o dalis DEC - išėjimus gr_{d1} ir Ce_{d1}. Šie išėjimai yra taip pat prijungti prie valdymo grandinės 27. Visi šie išėjimai dirba pirmosios taktinio ciklo pusės metu. Selektoriai įsijungia vėliau, bet irgi pirmoje takto pusėje, ir todėl selektorius gali nustatyti, ar inkrementoriaus ar dekrementoriaus grandinėje yra išėjimo signalai.

Normalus didžiausias užlaikymas pirmajam selektoriui sudaro daugiau kaip pusę taktinio ciklo. Bet šis įrenginys užlaikymo nesudaro. Vienok, bendru atveju normalus užlaikymas sudaro mažiau negu pusę ciklo. Nereikalingi pereinamieji procesai selektoriuose nesusidaro.

Žodžių selektoriaus M2 išėjimas n4 prijungtas tiesiai prie pirmojo žodžių selektoriaus M3, turinčio keturis įėjimus, įėjimo ir per invertorių INV prie antrojo šio selektoriaus įėjimo. Išėjimas

n2 yra eksponentinės dalies OP_e išėjimas. Įėjimas B_{bias2} yra prijungtas prie trečiojo žodžių selektoriaus M3 įėjimo.

Mantisės dalis OP_m turi penkis įėjimus, t. y. v_{2i}, v_{1i}, a_1, a_2 ir a_3. Visi šie įėjimai yra prijungti prie penkių žodžių selektoriaus M4, turinčio penkis įėjimus ir vieną išėjimą n8, įėjimų. Išėjimas n8 yra prijungtas prie eksponentinės dalies OP_e žodžių selektoriaus M3 ketvirtojo įėjimo. Išėjimas n8 taip pat yra tiesiai prijungtas prie vieno iš trijų žodžių selektoriaus M5 įėjimų, prijungtas per neigiamo postūmio įrenginį SH1, kuris perstumia dvejetainę informaciją savo įėjime per keturis žingsnius į mažesnio reikšmingumo pusę, t. y. dalina dvejetainę informaciją iš 16, antrojo žodžių selektoriaus M5 įėjimo, ir prijungtas per teigiamo postūmio įrenginį SH2, kuris perstumia dvejetainę informaciją savo įėjime per keturis žingsnius į didesnio reikšmingumo pusę, t. y. daugina dvejetainę informaciją iš 16, trečiojo žodžių selektoriaus M5 įėjimo.

Žodžių selektoriaus M5 išėjimas n7 yra tiesiai prijungtas prie pirmojo iš šešių žodžių selektoriaus M6 įėjimų; per neigiamo postūmio įrenginį SH3, kuris perstumia dvejetainę informaciją savo įėjime per vieną žingsnį į mažesnio reikšmingumo pusę, prijungtas prie antrojo įėjimo; per antrąjį neigiamo postūmio įrenginį SH4, kuris perstumia dvejetainę informaciją savo įėjime per vieną žingsnį į mažesnio reikšmingumo pusę, prijungtas prie trečiojo įėjimo; per teigiamo postūmio įrenginį SH5, kuris perstumia dvejetainę informaciją savo įėjime per vieną žingsnį į didesnio reikšmingumo pusę, prijungtas prie ketvirtojo įėjimo; ir per antrojo teigiamo postūmio įrenginį SH6, kuris perstumia dvejetainę informaciją savo įėjime per vieną žingsnį į didesnio reikšmingumo pusę, prijungtas prie penktojo įėjimo. Vidinis pastovaus žodžio generatorius 41, gavęs valdymo signalą įėjime $c2_{10}$, generuoja pastovų žodį c_{word2}. Šio generatoriaus išėjimas prijungtas prie atskiro žodžių selektoriaus M6 įėjimo. Pastovaus žodžio generatorius 41 gali generuoti nustatytas vienetų ir nulių kombinacijas, ar tam tikrus žodžius, atspindinčius tam tikrą informaciją. Tai valdoma signalo $c2_{10}$ pagalba. Generatorius gali saugoti tik ribotą kombinacijų, iš kurių signalo $c2_{10}$ pagalba pasirenkama tik viena, skaičių. Žodžių selektoriaus išėjimas yra mantisės dalies OP_m išėjimas n1.

Mantisės dalis OP_m vykdo sekančias funkcijas. Mantisės bazinė reikšmė n8 yra žodžių selektoriaus M4 pagalba išrenkama iš v_2, v_1, a_1, a_2 ar a_3. Ji gali būti naudojama tokia, kokia yra, ar paslinkta per 1, 2, 3, 4, 5, 6 bitus į kairę (teigiamas postūmis), ar paslinkta per 1, 2, 3, 4, 5, 6 bitus į dešinę (neigiamas postūmis) ar pakeista žodžiu pastovaus žodžio generatoriaus žodžiu c_{word2}.

Eksponentinė dalis OP_e vykdo sekančias funkcijas. Jos išėjime gali būti nepastumta mantisės reikšmė n8, v_2+1, invertuotos v_2+1, v_2-1, invertuotos v_2-1, $B_{co-incr2}$, invertuotos $B_{co-incr2}$, $B_{co-decr2}$, invertuotos $B_{co-decr2}$, B_{bias2}.

Operandų įrenginio r2 žodžių selektoriai M2÷M6 yra valdomi valdymo grandinės 27 išėjime c2 esančiu valdymo žodžiu. Taip, žodžių selektorius M2 valdomas valdymo žodžio dalimi $c2_3$, žodžių selektorius M3 valdomas valdymo žodžio dalimi $c2_2$, žodžių selektorius M4 valdomas valdymo žodžio dalimi $c2_6$, žodžių selektorius M5 valdomas valdymo žodžio dalimi $c2_5$, ir žodžių selektorius M6 valdomas valdymo žodžio dalimi $c2_4$. Taip pat, prijungti prie žodžių selektoriaus M4 išėjimo n8 postūmio įrenginiai SH1 ir SH2 yra valdomi dalies signalu $c2_8$, postūmio įrenginiai SH3÷SH6 yra valdomi dalies signalu $c2_7$, inkrementorius INC ir dekrementorius DEC valdomi dalies signalu $c2_9$, pastovaus žodžio generatorius 41, generuojantis žodį c_{word2} - $c2_{10}$, ir 0/1 generatorius 40 - signalu $c2_{11}$.

Bitų grupių individualus selektorius M1 yra valdomas sudėtiniu signalu, susidedančiu iš valdymo žodžio dalies $c2_1$, tikslumo dekoderio PD2 išėjimo kodo signalu B_{cde2}, paskutinio reikšmingiausio bito signalu $B_{1sb\ adj2}$, ir eksponentės signalu $B_{exp\ adj2}$. Paskutiniai du signalai gaunami iš skaičiavimo grandinių 42A ir 42B išėjimų B_{1sb2} ir B_{exp2}, ateinančių iš tikslumo dekoderio PD2. Šie signalai bus aprašyti žemiau. Grandinės 42A ir 42B valdomos signalo $c2_5$, kuris yra valdymo grandinės 27 išėjimo c2 dalis, pagalba. Tikslumo dekoderis PD2 gauna per magistralę v_{2i} eksponentės ženklą ir kodinę dalį, taip pat iš valdymo grandinės 27 gauna signalą $c5_3$, nurodantį ar magistralėje v_{2i} esantis žodis yra sveikas ar plaukiojančio kablelio. Valdymo signalas $c5_3$ yra ALU valdymo grandinės 27 išėjimo c5 dalis.

Visa tai galioja ir operandų įrenginio r1 tikslumo dekoderiui

PD1. Tikslumo dekoderio PD1 signalai B_{1sb1}, B_{exp1} yra apdorojami atitinkamose grandinėse 43A ir 43B su išėjimais $B_{1sb\ adj1}$ ir $B_{exp\ adj1}$. Grandinės 42A, 42B ir 43A, 43B atitinkamai perduoda dekoderio įėjimų informaciją į išėjimus. Kai kuriais atvejais, kai signalai gaunami iš inkrementoriaus INC ar dekrementoriaus DEC, reikalinga postūmio operacija, kas bus aprašyta toliau. Postūmis valdomas atitinkamais skaitmeninio ALU valdymo grandinės 27 išėjimų c2 ar c1 signalais $c2_5$ ar $c1_5$.

Tikslumo dekoderis PD2 (arba PD1) išėjimuose yra nereikšmingumo vertės, pvz. išėjimuose, prijungtuose prie grandinių 42A, 42B (arba 43A, 43B) yra vieni nuliai, kai žodis v_{2i} yra sveiko tipo, ir išėjimuose yra signalai, žemiau aprašyti kaip B_{exp2}, B_{1sb2}, ir B_{cde2} (arba B_{exp1}, B_{1sb1}, ir B_{cde}), kai žodis v_{2i} (ar v_{1i}) yra plaukiojančio kablelio tipo.

Tikslumo dekoderis PD1 išėjimuose generuoja signalus $B_{co\text{-}decr1}$, $B_{co\text{-}incr1}$, B_{bias1}, B_{exp1}, B_{1sb1}, B_{msb1}, ir B_{cde1}, kurie nueina į operandų įrenginį r1, ir signalus $B_{co\text{-}max1}$, indikuojantį, kad kodas atspindi maksimalų eksponentės ilgį, bei $B_{co\text{-}min1}$, indikuojantį, kad kodas atspindi minimalų eksponentės ilgį, kurie nueina į ALU valdymo grandinę 27. Tikslumo dekoderis PD2 išėjimuose generuoja signalus $B_{co\text{-}decr2}$, $B_{co\text{-}incr2}$, B_{bias2}, B_{exp2}, B_{1sb2}, B_{msb2}, ir B_{cde2}, kurie nueina į operandų įrenginį r2, ir signalus $B_{co\text{-}max1}$, indikuojantį, kad kodas atspindi maksimalų eksponentės ilgį, bei $B_{co\text{-}min1}$, indikuojantį, kad kodas atspindi minimalų eksponentės ilgį, kurie nueina į ALU valdymo grandinę 27. Tikslumo dekoderių PD1 ir PD2 atitinkami išėjimai B_{1sb}, B_{exp} turi būti perstumti tokiu pat būdu kaip ir mantisė, tam kad išėjimuose atsispinėtų naujas tikslumas. Perstūmimas yra valdomas iš signalų $c2_5$ arba $c5_1$ pagalba valdomų grandinių 42A ir 42B arba 43A ir 43B, kas aprašyta žemiau.

Tikslumo dekoderio išėjimo signalai smulkiau bus aprašyti žemiau.

Dauginimas ir dalinimas, operandų įrenginys r3

Trečias operandų įrenginys r3, daugybos/dalybos funkcijų įrenginys, yra kitokio tipo, negu įrenginiai r1 ir r2, ir naudojamas daugiausia daugybos ir dalybos operacijose. Įrenginys r3 daugybos ir dalybos operacijų metu vykdo postūmius ir užstatymus. Jis valdomas ALU valdymo grandinės 27 kontrolinio žodžio pagalba.

Daugybos/dalybos įrenginys r3 turi keletą įėjimų, kurių pirmasis gauna informaciją iš magistralės v_{2i}, antrasis - iš magistralės v_{3i} ir trečiasis - iš pastovaus žodžio generatoriaus 44, kuris pagal valdymo grandinės 27 pateiktą išėjimo c3 signalą $c3_4$ išduoda vieną iš kelių saugomų bitų kombinacijų.

Įrenginys r3 naudojamas sudėtinguose skaitmeninėse instrukcijose vieno argumento įvertinimui. Jo išėjimas yra prijungtas prie magistralės v_{3out}.

Įėjimas v_{2i} aptarnauja žodžių selektoriaus M11 pirmąjį įėjimą. Įėjimas v_{3i} yra tiesiogiai prijungtas prie žodžių selektoriaus M11 antrojo įėjimo, per bitų teigiamo postūmio įrenginį SH11, valdomo signalu $c3_5$, prie trečio įėjimo, per bitų neigiamo postūmio įrenginį SH12, valdomo signalais $c3_3$, B_{msb1}, N_{lsb1}, prie ketvirto įėjimo, ir per bitų neigiamo postūmio įrenginį SH13, valdomo signalais $c3_3$, B_{msb1}, N_{lsb1}, prie penkto įėjimo. Generatoriaus 44 išėjimas yra prijungtas prie šeštojo žodžių selektoriaus M11 įėjimo. Žodžių selektoriaus M11 išėjimas, prijungtas prie antrojo žodžių selektoriaus M12 įėjimo, yra tvarkomas ALU valdymo grandinės 27 valdymo žodžio dalimi $c3_2$.

M12 realizuotas kaip magistralinė jungtis tarp selektoriaus M11 ir išėjimo magistralės v_{3out}, ir valdomas valdymo žodžio dalimi $c3_1$.

Skaitmeninio ALU operacijų pavyzdžius galima surasti .....

Tikslumo dekoderis

Principinė tikslumo dekoderio struktūra yra parodyta pieš. 30. Reikia pažymėti, kad ši struktūra gali būti modifikuota įvariais tikslais, pvz. skaičiavimo greičiui padidinti. Grandinė, parodyta pieš. 30, gali turėti ir kitokią konfigūraciją, bet ji turi realizuoti tokias pačias principines funkcijas, kaip ir grandinė pieš. 30.

Įėjimo signalo v_{ji}, kur j yra 1 arba 2, eksponentės ženklas ir kodo dalis yra paduodama į dekodavimo grandinę 50, kuri savo išėjime pateikia vieną iš kodo, įjungiančio eksponentės ženklą, variantų priklausomai nuo įėjimo informacijos. Kitu atveju du variantai gali būti išduoti dekoderio 50 išėjime, bet visada ta

pačia tvarka. Išėjimas yra prijungtas prie bitų sekos komparatoriaus grandinės 51, kuri seka savo įėjime esančias bitų kombinacijas ir generuoja signalą B_{msb}, parodantį kodo bitų ilgį. Ji taip pat generuoja signalą B_{cde}.

Šešių bitų sekos komparatorių grandinės 52÷56 yra taip pat prijungtos prie dekodavimo grandinės 50. Kiekvienas iš komparatorių reaguoja į tam tikrą kodą, kurį gauna iš grandinės 50 išėjimo, savo išėjime generuodamas "1". Normaliai yra generuojamas "0". Tiktai grandinės 52÷57 savo išėjimuose gali užstatyti "1", kitos grandinės visą laiką duoda "0". Pirmasis ir paskutinis signalo B_{lsb} signalo bitai yra visada "0". Tokiu būdu, dvi linijos 58 ir 59 yra pastoviai prijungtos prie linijos, turinčios "0".

Linija 58, grandinių 52÷57 išėjimai ir linija 59 sudaro magistralę B_{lsb}, kurioje vienas bitas atspindi vieną tikslumo grupę. Magistralė B_{bias} turi keturis bitus vienai tikslumo grupei. Kiekviena grupė įjungia tris "0" bitus ir vieną signalo B_{lsb} bitą šiai grupei. Atitinkamai trys kiekvienos grupės linijos yra prijungtos prie "0", o ketvirtoji prie atitinkamo išėjimo B_{lsb} laido. Išėjimas B_{co_min} yra prijungtas prie grandinės 52 išėjimo, ir išėjimas B_{co_max} yra prijungtas prie grandinės 57 išėjimo.

Išėjimo B_{exp} bitų piešinys mažiausiai reikšmingoje dalyje turi "0" bitus iki pozicijos, nuo kurios prasideda išėjimo B_{lsb} "1", ir nuo čia visi bitai yra "1". Mažiausiai reikšmingas bitas linijoje 59 yra visada "0", bet jau sekančioje pozicijoje galimas "1". ARBA elemento 60 vienas įėjimas prijungtas prie grandinės 57 išėjimo ir kitas įėjimas - prie grandinės 56 išėjimo. Grandinės 57 išėjimas yra prijungtas prie signalo B_{exp} priešpaskutinės linijos. ARBA elemento 60 išėjimas yra prijungtas prie išėjimo B_{exp} sekančios linijos. Taip, jeigu grandinės 57 išėjime yra "1", tada ARBA elemento 60 išėjime irgi bus "1". ARBA elemento 60 išėjime bus "0" tik tada, kai tiek grandinės 57, tiek grandinės 57 išėjimuose bus po "0". ARBA elemento 60 išėjimas yra prijungtas prie kito ARBA elemento 61 pirmojo įėjimo. Kitas elemento 61 įėjimas prijungtas prie grandinės 55 išėjimo. ARBA elemento 61 išėjimas yra prijungtas prie trečios nuo galo išėjimo B_{exp} linijos ir prie ARBA elemento 62 pirmojo įėjimo. Antrasis ARBA elemento įėjimas yra prijungtas prie grandinės 54 išėjimo. Toks sujungimo būdas leidžia užstatyti "1"

visuose pozicijose, kurios yra to pačio ar aukštesnio reikšmingumo kaip išėjimo B_{1sb} pozicijos, turinčios "1".

Žodžio B_{co_decr} ženklo, eksponentės ženklo ir kodo dalies pirmoji linija prijungta prie įrenginio, generuojančio "1", sekanti linija - prie prijungto prie šeštos B_{1sb} linijos invertoriaus 161, trečia - prie ARBA elemento 162, kurio įėjimai prijungti prie penktos ir šeštos B_{1sb} linijų, ir penkta ir šešta linijos tiesiai prijungtos prie atitinkamai antros ir trečios B_{1sb} linijų. Žodžio B_{co_decr} ženklas, eksponentės ženklas ir kodo dalis prijungtos bitų komplektavimo grandinės 66, kuri prideda "0" gauname žodyje tam, kad suformuotų savo išėjime 32 bitų žodį, įėjimo.

Žodžio B_{co_incr} ženklo, eksponentės ženklo ir kodo dalies pirmoji ir antroji linijos prijungta prie įrenginio, generuojančio "1", trečioji linija - prie prijungto prie septintos B_{1sb} linijos invertoriaus 163, ketvirta - prie ARBA elemento 164, kurio įėjimai prijungti prie trečios ir ketvirtos B_{1sb} linijų, ir penkta linija tiesiai prijungta prie ARBA elemento 165, kurio įėjimai prijungti prie trečios ir penktos B_{1sb} linijų. Žodžio B_{co_incr} ženklas, eksponentės ženklas ir kodo dalis prijungtos bitų komplektavimo grandinės 67, kuri prideda "0" gauname žodyje tam , kad suformuotų savo išėjime 32 bitų žodį, įėjimo.

Sumatorius

Sumatoriaus, naudojamo grandinėse 28 ir 30 (pieš. 28), schema yrs parodyta pieš. 31A÷31C. Sumatorius turi du argumentus a ir b. Argumentas a gaunamas magistrale v_{1i} ir argumentas b gaunamas magistrale v_{2i}.Jie gali būti tiek sveiko, tiek plaukiojančio kablelio tipo. Sveiko skaičiaus atveju grandinės 27 išėjimo signalas $c5_1$ yra "0" ir magistralėse B_{1sb1} ir B_{1sb2} visi bitai yra taip pat "0", o plaukiojančio kablelio skaičiaus atveju visi šie signalai bus lygūs "1". Kaip matyti iš pieš. 31A. sumatorius yra dvejetainis sumatorius, padarytas iš kaskadiniu principu sujungtų sekcijų G0, G1, ..., Gi, ..., G7, t. y. viena sekcija skirta vienai tikslumo grupei.

Kaip matyti iš aukščiau aprašytų sudėties operacijų yra svarbu plaukiojančio kablelio atveju prieš sudėjimą išlyginti argumentus a ir b. Tai reiškia, kad B_{1sb1} ir B_{1sb2} turi būti lygios. B_{1sb1} signalo B_{1sb} pavidalu valdo sumatoriaus darbą. Signalas B_{1sb}

perstumiamas per vieną žingsnį mantisės link, kai jis yra prijungtas prie sumatorių, inkrementorių, dekrementorių ir t. t., kadangi jis turi reguliuoti (užblokuoti) labiausiai reikšmingos mantisės grupės perdavimą. Todėl mažiausio reikšmingumo bitas signale B_{lsb} tokiais atvejais yra nenaudojamas.

a ir b argumentai, kiekvienas iš 32 bitų, yra sudalinti į aštuonias grupes po keturis bitus. Kiekviena a ir b keturių bitų grupė paduodama į atskirą inividualios sekcijos Gi, kur i yra tarp 0 ir 7, įėjimą. Magistralės B_{lsb} kiekviena linija yra prijungta yra prijungta prie sekcijų Gi, išskyrus sekciją G0. Linijos c_{in}, ateinančios į sekciją G7, pagalba nurodoma ar sumatorius turi vykdyti sumavimą ar atėmimą. c_{in} yra "0", kai reikalingas sumavimas, ir "1", kai reikalingas atėmimas. Kiekviena sekcija generuoja signalą c_{out}, kuris nueina į sekančią sekciją, ir kiekviena sekcija gauna ateinantį iš ankstesnės sekcijos signalą c_{in}, kas bus smulkiau aprašyta žemiau.

Argumentų a ir b sumavimas yra vykdomas atskirai kiekvienoje sekcijoje nuo G0 iki G7, jeigu reikia perduodant signalą į sekančias sekcijas nuo G7 iki G0. Esant plaukiojančio kablelio sumavimui, mantisės perdavimui reikalui esant yra naudojama perdavimo magistralė c_m. Ši magistralė kartu su ženklo linija c_s ir eksponentės ženklo linija c_{se} sudaro grandinės 27 (pieš. 28) įėjimą gr_{a1} (arba gr_{a2} arba gr_{a3}). Grandinė 27 perduoda šią informaciją per magistralę A_{STATE} į valdymo įrenginį 6.

Signalas $c5_1$ parodo ar argumentai a ir b yra sveiko ar plaukiojančio kablelio tipo, signalas $c5_2$ nurodo ar kodas argumentui a yra normalus ar transformuotas, signalas $c5_3$ nurodo ar kodas argumentui b yra normalus ar transformuotas. Šie signalai paduodami į sekciją G0, kuri atitinka didžiausio tikslumo grupę, kuri įjungia reikšmingiausią bitų dalį (ženklinę dalį). Tai dalinai priklauso nuo operandų a ir b formato (ar jie abu yra sveiki ar plaukiojančio kablelio).

Trijų būsenų loginis elementas 70 yra naudojamas apibrėžtos reikšmės magistralėje c_m nustatymui, pvz. nustato "0", kai operandai yra sveiki skaičiai, ir atitinkamai mantisės perdavimas magistrale nevykdomas. Minėtas elementas 70 yra valdomas signalo $c5_3$ pagalba.

Sekcijos G1 schema parodyta pieš. 31B. Ji yra sudalinta į dalis AS1 - AS4. Sekcija įjungia pernešimo generatorių, kuris generuoja signalą c_{out_i} į sekančią sekciją. Atitinkamoje sekcijoje Gi kiekviena bitų grupė a_i ir b_i turi atitinkamus bitus a_{ik} ir b_{ik}, prijungtus prie sumatoriaus dalies ASk, kur k yra tarp 1 ir 4. Kiekviena sumatoriaus dalis išduoda išėjime sum vieną bitą sum_{ik}.

Kaip bus toliau aprašyta, komentuojant pieš. 31C, pernešimo generatorius gauna generavimo bitą Gen, indikuojantį, kad sekcijos suma yra didesnė arba lygi 16, perdavimo bitą Pr, indikuojantį, kad sekcijos suma yra didesnė arba lygi 15, pernešimo bitą c_{in_i} iš ankstesnės sekcijos, bitą c_{in_i4} iš sumatoriaus dalies AS4, ir bitą B_{lsb_i+1}.Bitas B_{lsb_i+1} per invertorių 75 paduodamas į trijų būsenų elemento invertuotą įėjimą, ir tiesiai paduodamas į trijų būsenų elemento 76 valdymo įėjimą. Bitas c_{in_i4} padudamas į antrąjį trijų būsenų elemento 76 invertuotą įėjimą. Taip, kai bitas B_{lsb_i+1} yra "0", tada trijų būsenų elementas 76 perduoda bitą B_{lsb_i+1}. Jeigu bitas B_{lsb_i+1} yra "1", tada trijų būsenų elementas 76 perduoda bitą c_{in_i4}, indikuojantį apie mantisės pernešimą, kai pastarasis bitas yra "1".

Mantisės pernešimas atliekamas tikslumo grupei, esančiai prie argumento eksponentinės dalies, ir tai vykdoma, kai šios grupės bitas B_{lsb_i+1} yra "1".

Pernešimo perdavimas iš sekcijos į sekciją yra blokuojamas tikslumo grupėje, esančioje prie eksponentinės dalies. Bitas B_{lsb_i+1} yra prijungtas prie IR loginio elemento 77 invertuoto įėjimo; prie kito šio elemento invertuoto įėjimo prijungtas signalas Gen, ir tokiu būdu elementas 77 perduoda signalą Gen tik tada, kai B_{lsb_i+1} yra lygus "0". Bitas B_{lsb_i+1} yra taip pat prijungtas prie IR loginio elemento 78 invertuoto įėjimo; prie kito šio elemento invertuoto įėjimo prijungtas signalas Pr, prie trečio šio elemento invertuoto įėjimo prijungtas bitas c_{in_i}, ir tokiu būdu elementas 78 perduoda signalą Pr tik tada, kai B_{lsb_i+1} ir c_{in_i} yra lygūs "0". ARBA loginio elemento 79 įėjimai yra prijungti prie IR loginių elementų 77 ir 78 išėjimų, ir elementas 79 išduoda išėjime "1" kai bent viename jo įėjime yra "1".

Kiekvienos sumatoriaus dalies ASk schema yra parodyta pieš. 31C. Įėjimai a_{ik} ir b_{ik} yra prijungti prie ARBA-NE loginio elemento 80

įėjimų, ir šis elementas išduoda "1", kai įėjimų reikšmės skirtingos, ir išduoda "0", kai jos sutampa. ARBA-NE loginio elemento 80 išėjimas prijungtas prie ARBA-NE loginio elemento 81, gaunančio antrajame įėjime bitą c_{in_ik}, pirmojo įėjimo. ARBA-NE loginio elemento 81 išėjime yra "1", kai įėjimų reikšmės skirtingos, ir išėjime yra sum_{ik}, kartu sutvarkant pernešimo bitą iš ankstesnios sumatoriaus dalies, arba, jeigu duota sumatoriaus dalis yra pirmoji, pernešimo bitą iš ankstesnės sekcijos.

Bitai a_{ik} ir b_{ik} patenka į ARBA elemento 82 įėjimus. Šio elemento invertuotas išėjimas prijungtas prie kiekvieno iš trijų IR loginių elementų 83, 84, 85 invertuoto pirmojo įėjimo. Invertuotas bitas c_{in_ik*} paduodamas į antrąjį IR loginio elemento 83 įėjimą, ir šis elementas išduoda bitą c_{in_ik}, jeigu bent vienas iš bitų a_{ik} ar b_{ik} yra "1". Invertuotas bitas P_{in_ik*}, ateinantis iš prieš tai esančios sumatoriaus dalies, paduodamas į antrąjį IR loginio elemento 84 įėjimą, ir šis elementas išduoda bitą P_{out_ik}, lygų P_{in_ik}, jeigu bent vienas iš bitų a_{ik} ar b_{ik} yra "1".Invertuotas bitas G_{in_ik*} paduodamas į antrąjį IR loginio elemento 85 įėjimą, ir šis elementas išduoda bitą G_{in_ik}, jeigu bent vienas iš bitų a_{ik} ar b_{ik} yra "1". Bitai P_{in_ik} ir G_{in_ik} yra "1" ir "0", kiekvienos tikslumo sekcijos pirmojoje sumavimo dalyje AS1 (pieš. 31B).

Bitai Pr ir Gen naudojami realizuojant vilnijantį pernešimą, t. y. pagreitintą pernešimo variantą. Kadangi pernešimo grandinėlės įėjimuose yra diskretiniai dydžiai, skaičiavimai tikslumo grupėse atliekami lygiagrečiai, t. y. vienu metu visoms tikslumo grupėms. Paskutinis pernešimas c_s turės tik 4+8 elementų užlaikymą, tuo tarpu nenaudojant vilnijančio pernešimo, susidarys 32 elementų užlaikymas.

Bitai a_{ik} ir b_{ik} taip pat patenka į IR elemento 88 įėjimus. Šio elemento invertuotas išėjimas prijungtas prie kiekvieno iš dviejų ARBA loginių elementų 86, 87 invertuoto pirmojo įėjimo. IR loginio elemento 83 išėjimas yra prijungtas prie ARBA loginio elemento 86, turinčio išėjimą c_{out_ik}, antrojo įėjimo. IR loginio elemento 85 išėjimas yra prijungtas prie ARBA loginio elemento 87, turinčio išėjimą G_{out_ik}, antrojo įėjimo.

Greito pernešimo grandinėlių realizavimui yra įmanoma gauti tik vieną užlaikymą, jeigu bitų dalis, generuojanti generavimo ir

sklidimo bitus, yra invertuota. Tokiu atveju kas antra tikslumo grupės sekcija turės invertuotus įėjimus (neparodyta).

Kaip matyti iš pieš. 31A kairės pusės, didžiausio reikšmingumo tikslumo sekcija G0 šiek tiek skiriasi nuo kitų sekcijų. Ji nenaudoja įėjimo argumento B_{lsb_0}. Tai yra gaunamas "0" grupės viduje. Vietoje to sekcija G0 gauna mantisės pernešimo signalą c_{in_m}. Jis naudojamas pernešimo signalui į ženklo bitą suformuoti. Išėjime yra vienas papildomas bitas indikuojantis eksponentės pernešimą. Pernešimas yra realizuojamas kaip ir kitose grupėse, išskyrus tai, kad B_{lsb_0} yra laikomas "0", t. y. neteisinga.

Vieno bito signalai $c5_1$, $c5_2$, $c5_3$ gali suformuoti sekančius bitinius piešinius: ($c5_1$, $c5_2$, $c5_3$)=(neteisinga, neteisinga, neteisinga), kas reiškia sveiką skaičių; ($c5_1$, $c5_2$, $c5_3$)=(neteisinga, neteisinga, teisinga), kas reiškia kad a ir b yra normalaus kodo; ($c5_1$, $c5_2$, $c5_3$)=(teisinga, neteisinga, neteisinga), kas reiškia kad a yra pakeisto, o b normalaus kodo; ($c5_1$, $c5_2$, $c5_3$)=(teisinga, teisinga, teisinga), kas reiškia kad a yra normalaus, o b pakeisto kodo.

Inkrementorius

Inkrementoriaus principinė schema labai panaši į sumatoriaus schemą. Pieš. 32 parodyta inkrementoriaus tikslumo sekcijos schema. Inkrementorius turi tik vieną įėjimą a' (neparodytas). Inkrementoriaus bitinės dalys AS1' - AS4' skiriasi nuo sumatoriaus dalių tuo, kad neturi generavimo bito Gen palaikymo. Dalies AS4' invertuotas išėjimas Pr ir invertuotas pernešimo signalas c_{in*} iš prieš tai esančios sekcijos yra prijungti atskirai prie IR loginio elemento 90 dviejų invertuojančių įėjimų. Pernešimo c_{in} nueinančio į pirmą sekciją G7 reikšmė "1" reiškia, kad turi būti įvykdytas inkrementavimas (padidinimas per vienetą). IR loginio elemento 90 išėjimas ir bitas B_{lsb_i+i} yra prijungti atskirai prie ARBA loginio elemento 92 dviejų invertuojančių įėjimų, ir šio elemento išėjime gaunamas pernešimo signalas c_{out} perduodamas į sekančią sekciją. Inkrementoriaus pernešimo išėjimas realizuotas taip pat, kaip ir sumatoriuje. Vilnijančio pernešimo grandinėlė c_{in}/c_{out} indikuoja ar suma yra didesnė ar lygi dviems. Vilnijančio pernešimo grandinėlė Pe' indikuoja ar suma yra didesnė ar lygi vienam.

Kaip ir sumatorius, inkrementatorius turi mantisės pernešimo

magistralę c_m, valdomą signalu B_{lsb_i+i}. Valdymo bitai c1g ir c2g nurodo, ar reikšmė yra sveiko ar plaukiojančio kablelio tipo. Valdymo bitai c1g ir c2g valdo trijų būsenų elementą kurio pagalba mantisės pernešimo magistralėje, kai ji nenaudojama, sugeneruojama tam tikra nustatyta reikšmė (neparodyta).

Dekrementeris turi tokią pačią konfigūraciją, kaip ir inkrementatorius, išskyrus tai kad jis turi invertuotą įėjimą a'_{in*} savo dekrementavimo dalyse, atitinkančiose inkrementatoriaus dalis AS1' - AS4'.

Selektoriai

Operandų įrenginių selektoriai yra realizuoti kaip perdavimo elementų eilutė. Selektoriaus išėjimas yra visada apibrėžtas. Jis turi valdymo žodžių sąrašą ir įėjimo reikšmes. Visi, išskyrus vieną, yra išjungti. Kiekvieno įėjimo valdymas realizuotas kaip komplementarus signalas. Visas selektorius valdomas tokių elementų sąrašo pagalba. Jie yra nustatomi į skirtingas būsenas, tam kad gautų signalą iš tam tikro nustatyto porto. Bazinio selektoriaus bito dalies įranga parodyta pieš. 33. Vienas kiekvieno įėjimo porto bitas nueinantis į selektorių S_{ina}, S_{inb}, S_{inc} ir S_{ind} yra prijungtas prie atitinkamų valdomų raktų S_a, S_b, S_c ir S_d. Valdymo laidų poros, atitinkamai S_{ca}, S_{cb}, S_{cc} ir S_{cd}, nueina į šiuos raktus. Valdymo laidų pora, leidžianti perdavimą per tam tikrą raktą, turi reikšmių kombinaciją "1,0", visos kitos poros - "0,1".

Reikia pažymėti, kad dauguma skaitmeninio ALU grandinių realizuotos pagal principus, kuriais remiasi sumatorius, ir prityrę specialistai lengvai gali pagal tuos principus sukonstruoti likusias grandines. Tokiu būdu yra nereikalinga toliau smulkiai aprašinėti atskiras grandines.

SKAITMENINIO ALU VALDYMO GRANDINĖS 27 APRAŠYMAS

Parodyta pieš. 28 skaitmeninio ALU valdymo grandinė 27 yra sudėtingos konfigūracijos loginių elementų visuma. Įtaiso pavidale ji negali būti pateikta, kadangi jos konfigūracija yra realizuota kompjuterio pagalba, keičiant algoritmus priklausomai nuo valdymo signalų ir informatyvių signalų. Skaitmeninių loginių elementų visuma realizuojama kaip skaitmeninės mikroschemos kaukės piešinys visam išradime aprašytam skaitmeniniam ALU. Tokiu būdu šią grandinę

neįmanoma detaliai parodyti. Įėjimų ir išėjimų ryšių valdymo grandinėje 27 sąrašas parodytas priede 2 (sveikų skaičių sandaugos atveju).

Priede 2 parodytas santykis tarp skaitmeninio ALU valdymo grandinės 27 įėjimų ir išėjimų kai vykdomas sveikų skaičių dauginimas.

Priedas yra sudalintas į skirtingus "kontrolinius uždavinius". Kiekvienas kontrolinis uždavinys atitinka vienam išėjimui, t. y. vienai mikroinstrukcijai.

ALU valdymo grandinės 27 išėjimai, kurie yra išvardinti po pavadinimu "mikroinstrukcijų linijos" atspindi signalų lygius magistralėse C1 - C5. Magistralės C1, C2 ir C3 yra padalintos į smulkesnes magistrales $C1_1$, $C1_2$, $C1_3$ ir t. t. Taip, įėjimai, išvardinti po pavadinimu "kontrolinis uždavinys 1" kaip "būsena, reiškiančiosios linijos - sąlyga 1" apspręs išėjimus po pavadinimu "mikroinstrukcijų linijos" žemiau. Įėjimai, išvardinti aukščiau kaip sąlyga 1, 2 ir 3 apspręs tuos pačius išėjimus kaip ir sąlyga 4.

Kiekviena sąlyga yra apsprendžiama kaip tam tikra kelių būsenų linijų signalų lygių (0/1) kombinacija, kai kitos linijos turi nepibrėžtas reikšmes (x - nesvarbu).

Visi priede parodyti įėjimo signalai yra pažymėti pieš. 28, išskyrus signalus co_limit1 ir co_limit2.

Pieš. 28 signalas co_limit1 yra padalintas į du komponentus B_{co_max1} ir B_{co_min1}.

Pieš. 28 signalas co_limit taip pat yra padalintas į du komponentus B_{co_max2} ir B_{co_min2}.

Išėjimas A_{STATE} yra tik visų grandinės 27 įėjimų kopija. Išėjimas A_{STATE} yra prijungtas prie valdymo įrenginio 6.

Reikšmė gali būti atspindima pirmos eilės redukuojančio procesoriaus bazinės ląstelės pagrindiniuose registruose ir kai kuriuose skaitmeninio ALU įrenginiuse kaip instrukcijos operand/mul_quotient arba polynome. Informacija, liečianti bitinės eilutės likusios dalies pobūdį, gali būti pateikta instrukcijos magistralėje arba kodo lauke.

PORTAS

Redukuojantis procesorius gali turėti vieną ar kelis portus duomenų įvedimui ir išvedimui į ir iš procesoriaus. Kiekvienas portas yra įrenginys leidžiantis procesoriui bendrauti su išorine įranga. Portas yra prijungtas prie transformuojančio interfeiso 7, esančio objektinėje atmintyje 1 (pieš. 1). Šiek tiek jį modifikavus, jis gali būti prijungtas prie duomenų perdavimo 4. Portą galima traktuoti kaip objektinės atminties išplėtimą, kadangi jame yra mažiausiai keturios atminties ląstelės su panašiomis funkcijomis ir struktūra kaip objektinės atminties ląstelės 10 (pieš. 2). Įėjimo/išėjimo operacijos porte vykdomos unifikavimo pagalba.

Vidinė elgsena

Objektinė atmintis relizuoja procesoriaus vidinę elgseną, viena elgsena kiekvienam portui irgi yra pritaikyta. Elgsenos semantinė prasmė yra laikinė struktūra, kuri gali būti atspindima kaip išraiškos struktūra. Laikinė struktūra gali būti nustatyta kaip proceso, sudaryto iš begalinės laikinių reikšmių sekos, būsena.

Išorinė elgsena

Įėjimo signalai į portus gali patekti iš išorinių procesoriaus atžvilgiu daviklių ir įrenginių, t. y. patekti iš išorinės aplinkos. Portas transformuoja šuos signalus į skaitmeninę formą pagal objektinės atminties formatą, t. y. į išraiškas. Tokio pobūdžio išraiškos yra laikinės struktūros, t. y. elgsenos, tokios kaip ir elgsenos procesoriuje. Procesoriaus atžvilgiu ši laikinė struktūra gali būti traktuojama kaip išorinė elgsena.

Įėjimas ir išėjimas per portą realizuojamas unifikavimo pagalba į abi puses, t. y. vidinę procesoriaus elgseną ir išorinę elgseną. Tiek vidinės tiek išorinės elgsenos aprašomos vienodai, kas leidžia procesoriui atlikti realaus laiko operacijas.

Išorinės elgsenos, t. y. laikinės sekos, pavyzdys yra pateiktas pieš. 34. Diskretizuojamų signalų seka gali keistis laike, t. y. kvantavimo periodai gali nesutapti, kaip matyti iš pieš. 34. Signalo seka čia parodyta kaip elementų porų sąrašas. Kiekviena pora įjungia laiko trukmę ir signalo dydį to laiko metu. Signalo dydžiai yra pažymėti q_i ir laikai t_i kur i yra reikšmės nuo 1 iki 6.

Interfeisas

Interfeisas turi turėti sekančias savybes:

1. Signalas yra matuojamas tam tikrais momentais. Išmatuotas signalas yra išlaikomas nekintamas iki sekančio matavimo.
2. Laikas turi būti skaičiuojamas arba pagal porto nustatomą fiksuotą ciklo dydį, arba pagal vykdomos programos nustatomą fiksuotą ciklo dydį, arba pagal kvantavimo intervalus apsprendžiamus programoje arba išorinio taktinio generatoriaus.
3. Išmatuotas signalas skaitmetinėje formoje įvedamas į mašiną.
4. Skaitmeninis signalas yra arba loginio arba sveiko/plaukiojančio kablelio reikšmės tipo.

Išradime aprašytas procesorius su gaunamomis reikšmėmis elgiasi skirtingai negu tradiciniai procesoriai. Jis traktuoja portą kaip interfeisą tarp procesoriaus programos ir procesoriaus aplinkos. Programa yra saugoma objektinėje atmintyje ir sukuria vidines elgsenas.

Programos vykdymas paremtas tuo, kad programa procesoriuje ir programa jo aplinkoje vienodai tvarko realaus laiko įvykius interfeise.

Įvedimas ir išvedimas į ir iš procesoriaus yra realizuojamas unifikavimo pagalba. Tai reiškia, kad laiko trukmė ir šios trukmės dydis yra interpretuojamas vienodai abiejose porto pusėse.

Programa gali nustatyti konkrečią laiko trukmę, arba ji gali jos nenustatyti, naudodama tam tikrą simbolį, pvz. $, pažymintį visas galimas reikšmes.

Programa gali nustatyti konkretų signalo dydį, arba ji gali jo nenustatyti, naudodama tam tikrą simbolį, pvz. $, pažymintį visas galimas reikšmes.

Taisyklinga programa turi palikti vieną iš alternatyvių elgsenų po unifikavimo operacijos, ir ši elgsena turi sutapti su elgsena porte. Jeigu nei viena vidinė elgsena neatitinka išorinės aplinkos atspindėjimui porte, unifikavimo operacija duos rezultate nothing (niekas), kas bus suprasta kaip programinė klaida, kadangi nothing atitinka prieštaravimą. Jeigu po unifikavimo operacijos lieka daugiau negu vienas elgsenos variantas, tai taip pat interpretuojama kaip programinė klaida.

Įėjimas ir išėjimas

Intervalas programoje yra aprašomas pavidale during t v, kur t yra laiko intervalo trukmė ir v signalo dydis.

Kai vidinė elgsena nustato konkrečią signalo reikšmę, taip pat vadinamą signalo dydžiu, ši reikšmė perduodama į išorę. Jeigu vidinė elgsena palieka signalo reikšmę nenustatytą, tam tikra signalo reikšmė yra įvedama iš išorės.

Kai vidinė elgsena nustato konkrečią laiko trukmės reikšmę, ši reikšmė priimama išorėje. Jeigu vidinė elgsena palieka laiko trukmės reikšmę nenustatytą, tam tikra laiko trukmės reikšmė yra užduodama iš išorės.

Porto įranga

Porto, skirto įvedimui ir/arba išvedimui schema parodyta pieš. 35. Portas per magistralę DU yra prijungtas prie interfeiso 7. Portas yra prijungtas prie pirmos eilės redukuojančio procesoriaus centrinio valdymo įrenginio 6, kuris ir valdo šio porto darbą. Portas įjungia identifikatoraus registrą P_{ID}, kuris nusako portą. Saugomas registre P_{ID} identifikatorius gali būti naudojamas kaip ir objektinės atminties 1 identifikatorius, t. y. gali būti panaudotas objektinės atminties ląstelės išraiškose. Šis identifikatorius naudojamas procesoriaus viduje. Portas taip pat įjungia identifikatoriaus registrą CE_{ID}, kuris gali būti naudojamas, kaip ir registras P_{ID}. Porte taip pat yra atminties ląstelės CE_{DU}, CE_{LAST}, CE_{NEXT}, kurios naudojamos kaip objektinės atminties ląstelės.

Kiekviena atminties ląstelė gali saugoti išraišką su mažiausiai trimis laukais, skirtais išraiškos trims elementams saugoti, pvz. during, trukmės laikas, ir signalo dydis.

Mažiausiai du paskutiniai paminėti išraiškos elementai gali būti saugomi reikšmė/nuoroda tipo laukuose. Išraiška during yra TYPE lauke. Kadangi portas visada vykdo during operaciją, lauko TYPE gali ir nebūti.

Įrangoje, parodytoje pieš. 35, during yra realizuota 'apply ('@) išraiškoje, patalpintoje karu su funkcijos vardu (trukmės kodu) tipo lauke. Funkcijos pavadinimas yra saugomas porto atminties ląstelės pirmame reikšmė/nuoroda lauke, kai ir visi fumkcijų pavadinimai (kaip + - * /), priklausantys tipui 'apply. Trukmės kodas gali

būti naudojamas kaip identifikatorius, pažymintis objektinėje atmintyje saugomos funkcijos apibrėžimą.

Identifikatorius id taip pat gali būti saugomas reikšmė/nuoroda lauke. Šis identifikatorius suriša porto struktūrą su trukmės struktūra, saugoma objektinėje atmintyje.

Jeigu laukas, saugantis identifikatorius, yra jau nebereikalingas, tada naudojami tik trys reikšmė/nuoroda laukai. Vienok tam, kad suderinti porto registrų struktūrą su kitų procesoriaus atminties ląstelių struktūra, turėtų būti įjungtas ketvirtas reikšmė/nuoroda laukas, pažymėtas kaip unused (nenaudojamas). Taip pat galima prjungti papildomą įrenginį per VALUE IN/OUT prie ketvirto reikšmė/nuoroda registro. Tai leis perduoti grupę reikšmių, pvz. trukmęir du signalo dydžio tipus.

Kiekviena porto atminties ląstelė gali turėti atributų (požymių) laukus. Atminties ląstelės CE_{DU}, CE_{LAST}, CE_{NEXT} tarpusavyje sujungtos magistralių $W_{@1}$, W_{I1}, W_{T1}, W_{V1}, W_{E1}, W_{T2}, W_{V2}, W_{E2}, kiekviena magistralė turi pvz.. 38 linijas ir yra prijungta prie kiekvieno registro. Indeksas I žymi magistralę, prijungiamą prie registrų trukmės kodo lauko, indeksas T žymi magistralę, prijungiamą prie registrų reikšmė/nuoroda lauko, žyminčio trukmę, indeksas V žymi magistralę, prijungiamą prie registrų reikšmė/nuoroda lauko, žyminčio signalo dydį, indeksas E žymi magistralę, prijungiamą prie registrų papildomo reikšmė/nuoroda lauko.

Indeksas 2 žymi magistralę prijungtą prie išorės, ir indeksas 1 žymi magistralę, prijungtą prie interfeiso 7.

Atminties ląstelės parodytoje porto schemoje saugo ankstesnę išraišką, dabartinę išraišką, ir sekančią išraišką, bei tolimesnių išraiškų identifikatorių. Atminties ląstelės turėtų būti fiksuotose vietose ir jų turinys gali būti sukeičiamas tarpusavyje. Vianas iš variantų gali būti tas, kad registrų ląstelės gali turėti tokią pačią struktūrą, kaip bazinės ląstelės 2 (pieš. 10) registrų ląstelės.

Galima tris pirmasias minėtas atminties ląsteles sutvarkyti taip, kad jos galėtų keisti vietą viena kitos atžvilgiu, pakeičiant jų pavadinimus, kas valdoma iš centrinio valdymo įrenginio 6.

Smulki sujungimų tarp atminties ląstelių schema nėra pateikta,

kadangi specialistai šioje srityje pagal bazinės ląstelės organizacijos principus gali realizuoti šių ląstelių schemotechniką. Valdymo realizavimas parodytas schematiškai, bet jis gali būti nupieštas kaip ir bazinės ląstelės reigstrų valdymo raktai.

Porte gali būti skirtingas atminties ląstelių skaičius, pvz. viena.

Laiko skaitiklis ir laiko komparatorius (sulygintojas)

Portas įjungia laiko skaitiklį T_{COUNT}. Jis valdomas taktinio signalo $CLOCK_p$, ateinančio iš centrinio valdymo įrenginio 6, pagalba ir matuoja laiką. Jis gali būti numestas arba išorinio numetimo signalo pagalba arba numetimo signalo iš valdymo įrenginio 6 pagalba. Laiko skaitiklis T_{COUNT} yra prijungtas prie pirmojo selektoriaus SEL_1, kuris valdant centriniam valdymo įrenginiui 6 prijungia laiko skaitiklio T_{COUNT} išėjimą arba tiesiai prie magistralės W_{T2}, prijungiant tokiu būdu prie atminties ląstelių laiko saugojimo laukų, arba prie laiko komparatoriaus T_{COMP} pirmojo įėjimo. Kitas laiko komparatoriaus T_{COMP} įėjimas yra prijungtas prie magistralės W_{T2}. Laiko komparatoriaus išėjimas yra išorinis laiko išėjimas. Laiko komparatoriaus išėjimas taip perduodamas į valdymo įrenginį 6.

Portas taip pat įjungia išorinės reikšmės įėjimą ir išėjimo porto laidą VALUE IN/OUT (REIKŠMĖ Į/IŠ). Jis yra prijungtas prie konverterio $CONV_{OUT}$, skirtą išėjimo signalui, pvz. tai gali būti kodas/analogas keitiklis. Jis taip pat prijungtas prie konverterio $CONV_{IN}$, skirto įėjimo signalui, pvz. tai gali būti analogas/kodas keitiklis, atliekantis įėjimo signalo diskretizavimą. Registravimo intervalai gali būti užduoti laiko komparatoriaus T_{COMP} išėjime arba tai gali būti numetimo signalai, nueinantys į laiko skaitiklį T_{COUNT}. Selektorius SEL_2, prijungtas prie magistralės W_{V2} ir valdomas prijungiamų atminties ląstelių CE_{DU}, CE_{LAST}, CE_{NEXT}, pagal valdymo signalą iš centrinio valdymo įrenginio 6 sutvarko ar signalo amplitudė pateks į portą, ar išeis iš porto. Valdymo įrenginys 6 pirmiausia nuskaito visų registrų tiek atminties ląstelėje, tiek išorėje turinį, nustatydamas ar jame yra simbolis $. Jeigu jis egzistuoja, atliekamas įvedimas iš išorės. Jeigu ne, tam tikra surasta reikšmė išvedama į išorę. Po to valdymo įrenginys 6 per porto selektorių atitinkamai sujungia registrus.

Galima padaryti ir labiau sudėtingesnį portą. Pvz. signalo amplitudės, patenkančios į VALUE IN/OUT gali būti konvertuojamos pagal formulę, saugomą signalo keitiklyje SC, prijungtame tarp selektoriaus SEL_2 ir magistralės W_{V2}. Konvertavimo funkcija gali būti pvz. integralų visuma, kur kiekvienas integralas skirtas tam tikrai signalų amplitudžių klasei.

Signalo amplitudės išvedimas

Išvedama informacija saugoma atminties ląstelėje CE_{DU}. Ši ląstelė turi informaciją pavidalo

during t v,

kur t yra trukmės laikas ir v yra signalo amplitudė. Signalo amplitudė šiame registre yra žinoma ir turi būti išvesta. Signalo amplitudė gaunama iš atminties ląstelės registre saugomos signalo amplitudės ir išvedama per selektorių SEL_2 ir konvertorių $CONV_{OUT}$.

Signalo amplitudės įvedimas

Atminties ląstelės CE_{DU} informacija yra

during t $,

kur t yra trukmės laikas ir $ yra visos galimos amplitudinės reikšmės, t. y. ši išraiška dirba su bet kuria pateikta signalo amplitude. Signalo amplitudė šime registre yra nepibrėžta, ir todėl registras yra paruošiamas gauti signalo amplitudę. Signalo amplitudė yra gaunama iš porto singalo amplitudės įėjimo/išėjimo, paverčiama į kodą A/D keitiklyje $CONV_{IN}$, ir patenka į atminties ląstelės CD_{DU} signalo amplitudės registrą.

Trukmės laikas nustatomas procesoriaus programos pagalba

Atminties ląstelės CE_{DU} informacija yra

during t v,

kur t yra trukmės laikas ir v yra amplitudinė reikšmė. Trukmės laikas yra žinomas. Laiko skaitiklis T_{COUNT} pradžioje yra numetamas iš centrinio valdymo įrenginio 6. Besikeičiančios laiko vertės skaitiklyje T_{COUNT} yra lyginamos su atminties ląstelės CE_{DU} laiko registre saugoma verte laiko komparatoriaus T_{COMP} pagalba. Laiko trukmė baigiasi, kai šios vertės tampa lygiomis.

Atminties ląstelė CE_{LAST} išvaloma ir atminties ląstelė CE_{NEXT} užpildoma per šią laiko trukmę. Pereinant prie sekančio ciklo, atminties ląstelės CE_{DU} turinys perkeliamas į atminties ląstelę

CE_{LAST} ir signalo amplitudė iš atminties ląstelės CE_{NEXT} pernešama į ląstelę CE_{DU}.

Trukmės laikas nustatomas išorės pagalba

Atminties ląstelės CE_{DU} informacija yra

during $ v,

kur v yra signalo amplitudė ir $ yra visos galimos laiko trukmės reikšmės, t. y. ši išraiška dirba su bet kuria pateikta laiko trukme. Tokiu būdu, laiko trukmė yra nežinoma ir išorė nustato laiko trukmę. Laiko skaitiklis T_{COUNT} pradžioje yra numetamas išorinio numetimo signalo pagalba. Besikeičiančios laiko vertės skaitiklyje T_{COUNT} yra įrašomos į atminties ląstelę CE_{DU}. Laiko trukmė baigiasi, kai ateina sekantis išorinis numetimo signalas. Tada išorinis numetimo signalas per selektorių SEL_1 atjungia laiko skaitiklį.

Atminties ląstelė CE_{LAST} išvaloma ir atminties ląstelė CE_{NEXT} užpildoma per šią laiko trukmę. Pereinant prie sekančio ciklo, atminties ląstelės CE_{DU} turinys perkeliamas į atminties ląstelę CE_{LAST} ir atminties ląstelės CE_{NEXT} turinys pernešamas į ląstelę CE_{DU}.

Pavyzdys

Porto operacijas iliustruojantis pavyzdys pateiktas žemiau. Sakykime, kad mes norime aptikti vieną iš skirtingų sekų, kur pirmosios sekos sekundinės signalo amplitudės yra apibrėžtos, pvz. yra lygios 17, ir visos laiko trukmės yra nustatytos, pvz 1 sekundė, ir antrosios sekos apibrėžtos visos amplitudinės reikšmės, pvz. 1, 2, 3, 4 ir t. t., bet nenustatyta laiko trukmė. Naudojamas portas yra vadinamas $port_1$. Išraiškos apibrėžimas:

unify($port_1$ "vidinė elgsena")

kur "vidinė elgsena" yra alt operacija, žyminti visas alternatyvines sekas, t. y. didelė duomenų struktūra sekančio tipo:

```
alt(  seq(during(1s 17)during(1s $)during(1s 17)....)
   seq(during($ 1)during($ 2)during($ 3)....))
```

Jeigu įėjimo signalo seka gali būti unifikuota aukščiau parodyta struktūra, įvedimas yra įvykdomas ir atliekami atitinkami veiksmai. Viena iš sekų yra ((1s 17)(1s 0)(1s 17)(1s 99)(1s 17)), kuri atitinka seką, kuri gali būti unifikuota vidinės elgsenos, pateiktos aukščiau, pirmos sekos pagalba. Kita priimama seka gali būti ((2s 1)(4s 2)(1s 3)), kuri unifikuojama vidinės elgsenos antrosios sekos

pagalba. Nepriimamos sekos pavyzdys gali būti ((1s 2)...), kadangi signalo amplitudė, t. y. 2, negali būti unifikuota nei vienos vidinės elgsenos sekos pagalba.

Ryšys su išore

Porto konstravimo metu buvo siekta padaryti portą, kuris perduoda ir priima signalus priklausomai nuo to ar signalas yra apibrėžtas ar ne, arba vidinė reikšmė yra apibrėžta ar ne (reikšmė $), neapsiribojant procesoriaus ir porto bendravimo sistema, aprašyta aukščiau. Todėl galima parašyti programą, kurioje portas yra nuskaitomas ir į jį įrašoma procesorius pagalba. Kaip pavyzdys yra sekanti programa:

```
        unify(port "vidinė elgsena")
turinti vidinę elgseną:

  alt( seq(during 1s 1)(during 1s 1)(during 1s 2)(during 1s 4)
    (seq(during 1s 1)(during 1s 1)(during 1s 3)(during 1s 9)
    (seq(during 1s 2)(during 1s 4)(during 1s 1)(during 1s 1)
    (seq(during 1s 3)(during 1s 9)(during 1s 4)(during 1s 4)
   ir t. t.
```

t. y. (seq(during įėjimas)(during išėjimas)(during įėjimas)(during išėjimas)

Šioje vidinėje elgsenoje išorė pateikia reikšmę ir procesorius sekančio intervalo metu grąžina šios reikšmės kvadratą. Šis metodas leidžia pilnai valdyti patenkančius duomenis.

Jeigu įėjimo reikšmės negali būti unifikuotos, t. y. negali būti padarytos adekvačios atitinkamoms reikšmėms procesoriuje, rezultatas bus nothing. Jis išplis iki operacijos alt ir to pasekoje bus pašalinta visa šaka iki alt išraiškos.

Vidinės elgsenos, aprašytos aukščiau, ilgis sutampa su įėjimo sekos ilgiu. Trumpesnis variantas parašytas, panaudojant funkcionalinės kalbos galimybes, atrodys taip:

```
seq(during(1s X)during(1s apply((lambda (arg) (* arg arg)) X))),
```

kur during(1s X) tvarko signalo amplitudės X įvedimą ir kur kitos during-struktūros, įjungiančios lambda išraišką, tvarko įėjimo reikšmės kvadrato išvedimą. Aprašyta struktūra gali būti bendresnės struktūros, tvarkančios daugiau įėjimų/išėjimų rekursyvinio

kreipimosi pagalba, dalis. lambda funkcija čia naudojama kėlimo kvadratu operacijai pažymėti.

H POARTAS

Įėjimo/išėjimo porto aprašymas gali būti labiau apibendrintas, siekiant realizuoti ir skirtingų nuo trukmės-struktūrų duomenų tipų išvedimą ir įvedimą. Šiuo atveju vietoje išorinės elgsenos pagalba atliekamo vidinės elgsenos unifikavimo mechanizmo, unifikuojančio laiko trukmes ir signalo amplitudes, gali būti naudojamas bitinių piešinių unifikavimas. Ateinantis signalas turi būti skaitmeninis. Ateinantis signalas gali būti duomenų struktūros ar programos struktūros, kaip H kalbos programos. Pvz. atminties ląstelės laukas porte gali būti unifikuotas bitinių piešinių pagalba.

Pavyzdys: vidinė struktūra apply(X1 X2) įėjime gali būti unifikuota bitiniu piešiniu ir rezultate gausis : apply(+ list(1 2)), kas yra struktūra parodyta pieš. 8A ir B. Čia X1 ir X2 atitinka "bet kuris bitinis piešinys". X1 yra unifikuojamas bitiniu piešiniu, atitinkančiu instrukcijos "+" kodą ir X2 unifikuojamas bitiniu piešiniu, atitinkančiu instrukcijos "list(1 2)" kodą.

Per H portą, kuris priima H kalbos kodą, programa gali būti pakrauta į objektinę atmintį. H portas taip pat gali būti naudojamas ryšiui tarp procesorių persiunčiant programas ar/ir duomenis. Perduotos programos, siekiant išvengti nedelsiamo jų vykdymo, pradžioje turi būti pažymėtos kaip duomenys.

PIRMOS EILĖS REDUKUOJANTIS PROCESORIUS

Trumpai reziumuojant, redukuojantis procesorius yra valdomas programos, turinčios tam tikrą, ne nuoseklią struktūrą, pagalba. Ši programa redukuoja minėtą struktūrą per tam tikrą redukavimo žingsnių skaičių. Redukavime naudojama medžių įvairūs minimizavimo tipai. Pirmos eilės redukuojantis procesorius įjungia objektinę atmintį 1, susidedančią iš daugelio ląstelių 10, saugančių informaciją, kurią galima redukuoti. Ryšio tinklas, atminties magistralės sistema ir išraiškos viršūnė yra valdomi centrinio valdymo įrenginio 6 pagalba, redukavimo rezultatai perduodami visoms atminties ląstelėms.

Ryšio tinklas yra magistralės sistema įjungianti valdymo ir duomenų linijas, kurios prijungtos prie minėtų atminties ląstelių.

Centrinis valdymo įrenginys 6 yra bendras visoms atminties ląstelėms. Iš atminties ląstelių 10 į centrinį valdymo įrenginį patenka informacija apie redukavimo tipą ir tada nustatomi reikalingi valdymo žodžiai. Kadangi centrinis valdymo įrenginys yra loginių elementų masyvas informacija iš atminties ląstelių į jo IR ir ARBA įėjimus patenka aukšto ir žemo lygio signalų pavidalu ir valdymo įrenginio išėjime gaunamas kompleksinis panašaus pobūdžio signalų rinkinys patenka į atminties ląstelių valdymo grandinių kontrolinius taškus.

Taip, kiekviena objektinės atminties ląstelė ir bazinė ląstelė gali saugoti visą informaciją, reikalingą redukavimo operacijai atlikti. Redukavimo informacija taip pat gali įjungti nuorodas į kitas atminties ląsteles. Kai vykdymas yra baigtas, t. y. struktūra minimizuota, rezultatas yra kanoninė išraiška. Jeigu kanoninė išraiška yra per didelė saugoti vienoje atminties ląstelėje, įtraukiomos nuorodos į kitas atminties ląsteles, kuriose saugomos likusios išraiškos dalys. Kaip pavyzdys, panaudotas įrangoje, yra penkių elementų sąrašas, saugojimui objektinėje atmintyje reikalaujantis dviejų atminties ląstelių.

Bazinė ląstelė gali atlikti visas redukavimo rūšis. Reikia pažymėti, kad bazinių ląstelių skaičius napribotas viena, jų gali būti ir keletas, kas schemoje neparodyta. Objektinės atminties ląstelės gali atlikti tik tam tikras kai kurių redukavimo operacijų dalis.

Kelios bazinės ląstelės gali būti naudojamos lygiagrečiam darbui su visa struktūra, egzistuojančia pilnų išraiškos medžių pavidale, arba su struktūros dalimis, egzistuojančiomis išraiškos medžių dalių pavidale. Kadangi redukavimo etapų rezultatas nepriklauso nuo redukavimo tvarkos, skirtingos bazinės ląstelės gali dirbti persidengiančiuose kontekstiniuose laukuose. Bendru atveju nenudojamos jokios duomenų vientisumo užtikrinimo priemonės - tai realizuojama procesoriaus darbo metu. Pvz., jeigu dvi bazinės ląstelės redukuoja dvi identiškas išraiškos medžio dalis, vienintelė pasekmė yra ta, kad padarytas vienas nereikalingas redukavimas - rezultatas yra toks pat.

Ryšio tinklas yra taip pat pritaikytas informacijos perdavimui tarp objektinės atminties išrinktos atminties ląstelės ir ryšio

tinklo išrinktos vienos iš bazinių ląstelių, bei tarp išrinktos bazinės ląstelės ir ryšio tinklo išrinktos vienos ar kelių objektinės atminties ląstelių.

Ryšio tinklas vykdo perdavimo operacijas tarp bazinės ląstelės ir bet kurios atminties ląstelės pakeisdamas gavėjo turinį perdavėjo turiniu. Dvipusis perdavimas vykdomas sukeičiant operacijoje dalyvaujančių ląstelių turinį. Kiekviena ląstelės išraiška, saugoma atminties ląstelėje įjungia vykdymo, pozicijos medžio struktūroje, identifikatoriaus, konteksto, reikšmių arba nuorodų į kitas ląsteles medžio aprašymus.

Reikšmių medis, įjungtas į daugelio lygių išraišką, turi galinius elementus ir sudėtinius elementus, kurie įjungia tipą ir reikšmių sąrašą. Išraiškos vykdymo pažymėjimas turi mažiausiai dvi būsenas, iš kurių pirmoji yra neveiklumo būsena ir antroji vykdomoji būsena. Išraiškos pozicijos medžio struktūroje pažymėjimas turi mažiausiai dvi būsenas, iš kurių pirmoji yra mazgo būsena ir antroji šaknies būsena.

Mažiausiai keli išraiškos pažymėjimai yra atspindimi skirtingomis būsenomis, kiekviena būseną sudaro keli bitai, sudarantys dvejetainį kodą.

Kiekvienas reikšmė/nuoroda elementas ląstelės išraiškoje yra sudarytas iš požymių žodžio ir skaitmeninio žodžio, kurių kiekvienas yra bitų kompozicija sudaryta iš bitų esančių 'teisinga' arba 'neteisinga' būsenoje. Požymiai yra skirstomi į tiesiogines ir netiesiogines klases. Skaitmeninis žodis gali būti pvz. sveiko arba plaukiojančio kablelio tipo. Sveikas skaičius yra saugomas elemente dvejetainiame formate, taip pat vadinamu atspindėjimu, taip, kad visos reikšmės eilėje nuo mažiausio iki didžiausio skaičiaus susideda iš kelių bitų reikšmių, ir kur nulis yra atitinka šios eilės vidurį, o jo dvejetainis formatas yra 'teisinga' reikšmingiausiame bite ir 'neteisinga' likusiuose bituose. Plaukiojančio kablelio skaičius yra saugomas elemente dvejetainiame formate, taip pat vadinamu atspindėjimu, ir įjungia ženklą, eksponentės ženklą ir kodo lauką, eksponentės lauką ir mantisės lauką, minėti ženklas, eksponentės ženklas ir kodo laukas indikuoja į dalinimo ribą tarp minėto eksponentės lauko ir mantisės lauko, ir todėl eksponentė ie mantisė turi varijuojančius ilgius.

Aprašytas išradime procesorius naudoja minėtų skaitmeninių reikšmių kodavimą į dažnuminį atspindėjimą, t. y. kiekvienas užkoduotas reikšmės atspinėjimas atitinka tik vienai interpretuojamai reikšmei. Pirmasis kodavimo tipas yra naudojamas dvejetainiuose žodžiuose, atspindinčiuose sveiką tipą. Antrasis kodavimo tipas yra naudojamas dvejetainiuose žodžiuose, atspindinčiuose plaukiojančio kablelio tipą. Kodavimas yra toks, kad po kodavimo plaukiojančio kablelio tipas atitinka sveiką tipą. dažnuminis atspindėjimas pasiekiamas įvedant virtualų "1", t. y. "1" realiai fiziškai neegzistuoja. Skaitmeninių reikšmių atspindėjimai yra sudalinti į bitines grupes, vadinamas tikslumo grupėmis, turinčias nustatytą bitų skaičių, eksponentinės dalies ir mantisės dalies padalinimas, kai atspindėjimas atitinka plaukiojančio kablelio reikšmę, realizuoja skyrimą tarp dviejų tikslumo grupių.

ANTROS EILĖS REDUKUOJANTIS PROCESORIUS (SOP)

Kaip parodyta pieš. 36, tam tikras pirmos eilės procesorių FOP1 - FOPN skaičius gali būti sujungtas tarpusavyje tinklo NET pagalba, taip sujungti pirmos eilės procesoriai sudaro antros eilės redukuojantį procesorių.

Prijungimas prie tinklo NET gali būti realizuotas kaip duomenų perdavimo priemonė, kaip priemonė 4 pieš. 1, kiekviename pirmos eilės redukuojančiame procesoriuje FOPi, kur i yra nuo 1 iki N. Duomenų perdavimo priemonės 4 pieš. 1 įjungia regisrtrų R_{id}, R_{env}, R_{v1}, R_{v2}, R_{v3} eilutę. Kiekvienas tinklo registras prijungtas prie vertikalios magistralės linijos, turinčios to registro indeksą. Tinklo registrai yra naudojami, kai kelios objektinės atmintys kartu su bazinėmis ląstelėmis yra naudojamos lygiagretiems skaičiavimams. Ryšys tarp visų šių objektų palikomas tinklo registrų pagalba.

Teoriškai visi pirmos eilės redukuojantys procesoriai dalinasi ta pačia informacija M, kuri priklauso visos sistemos sričiai. Reikalinga, kad visų pirmos eilės procesorių visuma būtų lygi sistemos sričiai M. Tačiau kiekvienas pirmos eilės procesorius dirba su informacija, saugoma jo objektinėje atmintyje. Tai reiškia, kad paprasta išraiška gali rastis daugiau negu viename procesoriuje. Kiekvienas procesorius gali saugoti kelią, kuriame nurodoma, kuris pirmos eilės redukuojantis procesorius saugo išraišką. Randamos

keliose objektinėse atmintyse paprastos išraiškos yra identiškos - tai yra užtikrina redukavimo mechanizmas.

Išraiškos tinklas įjungia išraišką ir jos tėvus. Yra galima pernešti išraiškų tinklus pernešant išraiškas tarp procesorių. Komunikavimo minimizavimui pasiekti yra reikalinga semantiškai surištas išraiškas saugoti sukauptas viename ar keliuose procesoriuose. Bendravimo protokolas turi palaikyti išraiškų pernešimą tarp procesorių ir tada, kai kitas bendravimas tarp procesorių yra nenaudojamas. Išraiškų medžiai gali būti redukuojami lygiagrečiai skirtinguose procesoriuose. Šioms operacijoms yra nereikalinga sinchronizacija. Išraiškos medžių dalių redukavimas gali būti atliekamas skirtingu metu. Tai neiššaukia klaidos, ir redukavimo rezultatas bus tas pats, neatsižvelgiant į redukavimo tvarką.

Paprastai išraiškos, turinčios identiškas kopijas kituose procesoriuose, redukavimas vyksta vyksta tik viename FOPi (i yra tarp 1 ir N) iš lygiagrečių procesorių FOP1 - FOPN, ir redukavimo rezultatas, kai tik yra gaunamas, išplatinamas į visus procesorius, prijungtus prie tinklo NET ir tokiu būdu sutaupomi procesoriaus resursai. Vienok žymiai efektyviau ir trumpiau yra vykdyti redukavimą lygiagrečiai skirtinguose pirmos eilės procesoriuose. Pvz., tinklas NET gali būti užimtas kelių pirmos eikės procesorių aptarnavimu, kol jie atlieka lygiagretaus redukavimo operacijas.

Tinklas NET nėra išradimo dalis ir todėl nebus detaliai aprašytas. Yra pateikti keli tinklo variantai. Egzistuoja aibė skirtingos ideologijos tinklų, kurie gali būti panaudoti. Tinklas NET gali būti žiedinis su viengubais ar dvigubais ryšiais, žvaigždinis tinklas, pilnai sujungtas tinklas ir t. t.

Tam kad sumažinti ryšių skaičių tarp pirmos eilės procesorių FOP1 - FOPN, duomenų perdavimas tarp jų gali vykti nuosekliai nuoseklios magistralės pagalba. Duomenų perdavimas galimas išsklaidyto valdymo pagalba. Tada nenaudojamas centrinis valdymo įrenginys, centralizuotai tvarkantis pirmos eilės procesorių, žemiau vadinamų FOP, darbą. Vienas iš FOP valdo tinklą NET per magistralę DEM, o kiti FOP nustatomi į gavimo būseną. Signalų perdavimas užimant liniją ir tam panaudojant atskirą magistralę, gali būti realizuotas ir be atskiros magistralės, pvz. siunčiant tuščią signalą duomenų magistrale, kas aprašyta paraiškoje Nr. ..... Siunčiami tarp pimos

eilės procesorių duomenys gali būti saugomi registrų duomenų pernešimo priemonėse, kol šiuose FOP vyksta redukavimo operacijos. Duomenų perdavimo priemonės yra valdomos nuosekliai FOP centrinio valdymo įrenginio pagalba.

FOP sinchronizacija yra atliekama sinchronizuojant taktinius signalus tarp FOP. Toks sinchronizacijos suderinimo būdas yra plačiai naudojamas ir todėl nebus smulkiai aprašytas.

Galimas ir pirmos eilės procesorių ryšio palaikymo variantas, panaudojant centrinį valdymo įrenginį, valdantį FOP'ą bendravimą (neparodyta). Šis centrinis įrenginys realizuojamas kaip loginių elementų masyvas ir dirba su visais pirmos eilės procesorių valdymo įrenginiais. Tačiau tokio papildomo valdymo įrenginio panaudojimas yra nepageidautinas.

Pieš. 36 parodyta labai bendra išradime aprašyto antros eilės redukuojančio procesoriaus schema, kuriame duomenų perdavimas tarp FOP'ų gali vykti bet kokiu būdu.

Pirmoji antros eilės redukuojančio procesoriaus (SOP) schema

Pagal pieš. 37A parodytą antros eilės redukuojančio procesoriaus schemą, kuri skirta duomenų perdavimui tarp FOP'ų, keletas FOP'ų, $FOP'_{1,1}$ - $FOP'_{M,M}$, gali būti sujungti į kvadratinę matricą. Kiekvienas minėtos matricos FOP'as įjungia duomenų perdavimo kanalą CAN, prijungtą prie kiekvieno kaimyninio pirmos eilės procesoriaus šioje matricoje. Kaip parodyta pieš. 37B atskiros perdavimo priemonės DTF1 - DTF4 yra skirtos kiekvienam perdavimo kanalui CAN, ir kiekvienam prijungtam prie FOP'o portui. Tai leidžia vienu metu perduoti duomenis į kelis kitus FOP'us.

Struktūroje, parodytoje pieš. 37A, kiekvienas FOP'as bendrauja daugiausiai su keturiais kaimyniniais FOP'ais atskirų kanalų pagalba. Reikalingas tik paprastas valdymas tarp kiekvienos ryšių priemonių poros, kas pasiekiama užstatant komunikacinės linijos pareikalavimo žymę, tuo tarpu kai kiti FOP'ai paruošiami gauti duomenis per savo ryšio priemones DTFn.

Išraiškų tinklai ir FOP'ų sritys

Išraiškos gali būti priskirtos skirtingiems FOP'ams. Kaip aprašyta aukščiau, ląstelės išraiška yra pažymima mašininiu identifikatoriumi, pvz. identifikatoriumi #5. Ląstelės išraiška gali turėti

netiesioginius elementus, identifikatorius, surišančius ląstelę su kitomis ląstelės išraiškomis. Visos ląstelės išraiškos įjungiančios identifikatorius, nurodančius į tam tikrą išraišką, ir ši tam tikra išraiška suformuoja išraiškų tinklą, kuris įjungia išraiškas ir jų tėvus. Išraiškų tinklas yra tam tikroje FOP'ų srityje. Sritis paprastai yra iš FOP'ų sudaryta kvadratinė matrica, tai parodyta pieš. 37 apibrėžta ištisine linija. Ši sritis įjungia visas išraiškas su identifikatoriumi #5. Ląstelės išraiška su identifikatoriumi #5, prie kurios pririštos kitos išraiškos su identifikatoriais #5, gali įjungti identifikatoriaus elementus, kaip #2 ir #9, kurie randasi FOP'ų srities išorėje. Plačiau apie regionus bus paaiškinta žemiau, aprašant pieš. 37C.

Vienas iš sričių panaudojimo variantų yra tas, kad gali būti apribota asociatyvinė paieška ir pakeitimas, apribojimas ta prasme, kad šios operacijos gali būti atliekamos tik sričiai priklausančiuose FOP'uose redukavimo rezultatų realizavimui .

Bendravimas su išore

Tam tikras įėjimo/išėjimo portų PORT skaičius duomenų perdavimui į ir iš antros eilės procesorių naudojamas, patalpinant juos mažiausiai viename FOP'ų kvadratinės matricos krašte. Kiekvienas portas yra įrenginys, kurio pagalba vykdomas bendravimas su išoriniais įrenginiais. Tam gali būti panaudotas aukščiau aprašyto tipo portas, atliekantis įvedimo/išvedimo operacijas unifikavimo pagalba.

Per kitą portą - H-portą, kuris 'supranta' H kalbos kodo formatą, programos gali būti pakraunamos į FOP'ų objektinę atmintį. H portas taip pat gali būti naudojamas ryšiui tarp procesorių persiunčiant programas ir/arba duomenis.

Dar kitas porto tipas gali būti naudojamas ASCII teksto gavimui, kas leidžia išradime aprašytam procesoriui bendrauti su įrenginiais, galinčiais priimti ar siųsti ASCII formato tekstus, t. y. su tradiciniais kompjuteriais. Šis porto tipas nepriklauso išradimui ir todėl nebus smulkiau aprašytas.

Programiniai ir aparatūriniai žiediniai ryšiai

Aparatūrinis žiedinis ryšys yra aparatūriškai realizuotas bendravimo kelias. Programinis žiedinis ryšys yra kelias nustatytas tam tikrame abstarkčiame lygyje ir gali būti modifikuotas laikui

bėgant. Yra suprantama, kad programiniai žiedai ryšiui naudoja aparatūrinę dalį. Kelias, įjungiantis procesorius, kurie turi tam tikrą išraiškų tinklą, pvz. su identifikatoriais #5, yra vadinamas programiniu žiediniu ryšiu.

Išraiškų tinklo sritis yra visų procesorių poaibė. Išraiškų tinklas, t. y. išraiškos ir jų tėvai, yra šioje srityje. Norint sumažinti perdavimo operacijų skaičių tarp procesorių, reikalinga minimizuoti kiekvieno išraiškų tinklo regioną.

Tiktai programiniai žiedai

Sritis gali būti sudaryta, kaip parodyta pieš 37D, iš kelių FOP'ų (pieš. 37A) turinčių išraiškas su identifikatoriais ID_{def} saugomais skirtingose savo objektinės atminties vietose (pažymėta maža horizontalia linija pieš. 37D). Jeigu išraiška turi būti pakeista, tada visi FOP'ai turintys šios išraiškos identifikatorių (tėvus) atnaujinami programinio žiedo www_1, įjungiančio FOP'ą su identifikatoriumi ID_{def}, pagalba. Programinio žiedo dydis priklauso nuo nuo įrangos. Schemoje, parodytoje pieš. 37D programinis žiedas, sujungiantis FOP'us esančius išraiškų tinkle, yra serpantino formos. Tiktai programinio ryšio žiedai yra skirtingi skirtingiems išraiškų tinklams.

Sričių programiniai žiedai

Kaip grynai programinio ryšio žiedams, kur išraiškų tinklas ir jos programinis žiedas įjungia tuos pačius FOP'us, galimas variantas su programiškai realizuotu žiediniu ryšiu, įjungiant FOP'us, nenaudojamus atitinkančiame išraiškų tinkle. Programinis žiedas gali įjungti kiekvieną srities FOP'ą. Pieš. parodyta sritis, ir demonstruojama, kad pertvarkymo pranešimas siunčiamas į kiekvieną išraiškų tinklo www_2, suformuoto identifikatoriaus ID_{def} pagalba (t. y. programinis žiedas įjungia kiekvieną srities FOP'ą), srities FOP'ą. Regioniniai programiniai žiedai (www_2) yra labiau naudotini, kadangi sąnaudos yra mažesnės negu naudojant tiktai programinius žiedus (www_1). Srities programinis žiedas tam tikrai sričiai yra tas pats kiekvienam išraiškų tinklui srities viduje.

Bazinė regioninių tinklų forma yra magistralių žiedai. Žiedas turi pernešti reikšmę per vieną apsisukimo ciklą. Tokia operacija atitinka magistralės vieną taktinį ciklą.

Žiede keli išraiškų redukavimai gali būti pasiųsti vienu metu. Kai toks pranešimas apeina vieną ratą, žiedas turi jį pašalinti.

Antroji antros eilės redukuojančio procesoriaus (SOP) schema

Pagal antrąją schemą, parodytą pieš. 38, keletas FOP'ų yra sujungti į hierarchinį tinklą, kur kiekvienas trinklas yra magistralė NET_1 - NET_N. Pagrindinis skirtumas nuo schemos, parodytos pieš. 37A, yra tas, kad pieš. 37A FOP'as yra prijungtas per duomenų perdavimo kanalą prie daugiausiai keturių kitų FOP'ų, kai tuo tarpu pieš. 38 keli FOP'ai yra sujungti tarpusavyje ryšio magistralės sistema. Magistralės valdymas gali būti realizuotas, įvedant paprastą prioritetų dekoderį (neparodytas) kiekvienai magistralei, tam kad nustatyti prioritetus tuo atveju, kai daugiau kai vienas FOP'as pareikalaus duomenų perdavimo į kitus procesorius. Šis prioritetų dekoderis gali turėti beveik tokią pačią konfigūraciją, kaip parodytas pieš. 21A ir 21B (tai yra prioritetų dekoderis 11 pieš. 2 ir 3).

Kai viename FOP'e įvyksta redukavimas, redukuotos išraiškos tėvus reikia atnaujinti. FOP'as, atlikęs redukavimą, pasiumčia pertvarkymo pranešimą į magistrales, t. y. į magistralių NET_1 - NET_A poaibę, priklausomai nuo išraiškų tinklo srities. Visi prie FOP'ai, prijungti prie magistralės, kuria siunčiamas šis pranešimas, pasitikrina ar turi išraiškų tinklo dalį, ir jegu turi, ši dalis yra perrašoma/atnaujinama. Kai redukavimas patenka į magistralę jis taip pat pažymimas kaip unfikacija lokaliniuose FOP'uose. Kadangi keli FOP'ai atlieka redukavimą vienu metu, pakeitimo pramešimai turi būti paskirstomi pagal tam tikrą bendravimo protokolą. Kadangi rezultatas yra nesurištas su redukavimo vykdymo tvarka, magistralės bendravimo tvarkos apibrėžimas yra nereikalingas. Bendri komunikavimo protokolo principai gali būti pnaudoti užtikrinant, kad vienu metu perduotų tik vienas FOP'as.

Portai $PORT_2$, $PORT_{31}$, $PORT_{32}$ gali būti prijungti pasirenkant atitinkamas magistrales NET_2 ir NET_3. Kaip parodyta, galima prijungti kelis portus prie tos pačios magistralės $PORT_{31}$, $PORT_{32}$.

Trečioji antros eilės redukuojančio procesoriaus (SOP) schema

Pagal trečiąją schemą, parodytą pieš. 39, keletas FOP'ų FOP' yra sujungti į hierarchinį tinklą, kur kiekvienas tinklas yra

aparatūrinis ryšio žiedas nuo RING1 iki RING4, t. y. žiedas kur FOP'ai kietai sujungti tarpusavyje. Sritis gali perdengti vieną ar kelis žiedus. Programinio ryšio žiedai gali sutapti su aparatūriniais žiedais.

Skirtingų SOP'ų rūšių kombinacijos

Yra įmanoma, kaip parodyta pieš. 40, kombinuoti schemas, parodytas pieš. 37A, 38 ir 39 sujungti keletą antros eilės redukuojančius procesorius i hierarchinį tinklą, turintį mažiausiai dvi tinklo rūšis, iš kurių pirmoji yra magistralinis tinklas BUSNET, antroji - žiedinis tinklas RINGNET ir trečioji - matricinis tinklas SQUARENET.

Antros eilės mašina, pagal išradimą, gali turėti magistralinius ir/arba žiedinius tinklus. Kai išraiškų medžio dalis yra redukuota, redukuotų sūnų tėvai turi būti informuoti, kad sūnus yra redukuotas. Jeigu taip atsitinka, yra gerai, kai sūnus ir jo tėvai yra patalpinti į FOP'ų žiedą ar magistralę, ir tada informacija, liečianti sūnaus redukavimą, yra pasiunčiama per žiedą ar magistralę. Mašinos protokolas yra toks, kad išraiškos - tėvai yra perkeliami tarp FOP'ų, siekiant padaryti jas pasiekamas sūnui ar sūnums, t. y. išraiškų medžio dalys turėtų būti apjungtos į mažas sritis.

Mašininis protokolas

Tinkamas protokolas turi konfigūracinę dalį, kurioje apibrėžiamas portų sąrašas ir jų paskirtis. Išradime aprašyto procesoriaus portai, ar jie yra pirmos eilės procesoriuje, ar antros eilės procesoriuje ar aukštesnės eilės procesoriuje, gali būti panaudoti tam tikroms funkcijoms.

Pateikiamas protokolas įjungia vieną dalį, nustatančią portų konfigūraciją, vieną dalį pateikiančią FOP'ui uždavinius, vieną dalį, aprūpinančią redukavimo aritmetiką, vieną dalį, aprūpinančią pakankamai ilgus sąrašus, ir vieną dalį aprūpinančią pakankamai ilgų bitinių laukų reikšmes. Protokolo dalys gali būri realizuotos aparatūriškai. Šis protokolas sutvarko taktinės sinchronizacijos inicializavimą FOP'uose. Tai yra daroma tam tikrais laiko intervalais, pakankamai trumpais, kad būtų galima atlikti sinchronizavimą.

Elgsena yra atspindima kaip išraiška. Pakrovimo procedūra yra·

valdoma tam tikros protokolo dalies. Išraiškos pakrovimas ar perkrovimas gali vykti gana dažnai. Bet kuris FOP'as gali nurodyti pernešti išraišką, apklausiant kitus FOP'us, ar juose gali būti įrašyta pernešama išraiška.

Kai išraiška yra pernešama, pvz. kai FOP'o objektinės atmintis yra pilna, FOP'as kurieme yra išraiška siunčia paklausimą į magistralę ar žiedą, užklausdamas kitus FOP'us ar išraiška gali būti permešta į šiuos FOP'us. Jeigu visi šie FOP'ai atsako neigiamai, kas suprantama kaip, kad jie negali priimti šios išraiškos, išraiška yra paliekama pradiniame FOP'e. Jeigu vienas iš užklaustų FOP'ų atsako teigiamai, išraiška pernešama į šį FOP'ą. Jeigu keli FOP'ai atsako teigiamai, proritetų dekoderis išrenka vieną FOP'ą, į kurį ir bus pernešta išraiška. Prioritetų tvarka gali būti organizuota topologiniais principais, kaip pvz. arčiausio kaimyno principu.

Sritys

Pieš. 37C parodytas antros eilės procesoriaus kvadratinis laukas, susidedantis iš 256 FOP'ų. Kanalai tarp FOP'ų ir galimų portų į antros eilės procesorių neparodyti. Kiekvienas FOP'as atitinka nulinio dydžio sritį. Pieš. 37C parodyta, kad antros eilės procesoriaus matrica gali būti sudalinta į skirtingo dydžio logines (ne fizines) sritis, patalpintas viena šalia kitos taisyklinga tvarka. Kiekviena sritis yra 2^n+2^n FOP'ų dydžio. Sritys nepersidengia. Vienok protokolas gali būti toks, kad regionas gali būti perkeltas per pusę regiono dydžio per eilutes ir/arba stulpelius.

Protokolas saugo informaciją apie sričių vietą, ir žino kurioje srityje ar srityse randasi tėvas ar tėvai, ir naudoja tai ieškant ir pakeičiant redukuotą išraišką. Tai yra relizuojama per netiesioginius elementus, t. y. identifikatorius, esančius reikšmė/nuoroda lauke, įjungiant bitinį lauką, kuriame nurodomas srities dydis ir įjungiant dalį, kurioje nurodomas srities perkėlimas. Perkėlimo krypties bitai gali būti du, iš kurių vienas, kai yra "1", nurodo kad sritis yra perkelta į dešinę, ir kitas, kai yra "1", nurodo, kad sritis yra perkelta į viršų. Kai abu bitai yra "1", tai reiškia, kad sritis yra perkelta į viršų ir į dešinę.

Kai sūnaus redukavimas baigtas, informacija apie tai yra pasiunčiama per jo sritį(is), kur randasi jo tėvai, programinio

ryšio žiedo, įjungiančio visus tos srities(čių) FOP'us, pagalba. Tai reiškia, kad kai sritis yra labai didelė, programinio ryšio žiedas irgi yra didelis, ir informacija apie redukavimą užims daug laiko ir mašininių resursų. Tuo atveju protokolas yra toks, kad sutraukia išraiškų tinklo sritis kiek tai yra įmanoma. Jeigu atstumas tarp sūnaus ir tėvo yra didelis, mašininis protokolas duoda didesnį prioritetą tokioms siunčiamoms išraiškoms, negu išraiškoms, kurios perduodamos mažesniu atstumu. Reikia pažymėti, kad sutraukimas turi savo ribas. Kai kurios objektinės atmintys gali pavyzdžiui būti pilnai užpildytos duomenimis. Kai kurių tėvų sūnūs gali būti pakankamai toli nuo sutraukiamų sričių.

Kvadratinės matricos antros eilės procesoriaus protokolas turėtų būti toks, kad semantiškai surišti duomenys būtų mažoje srityje. Taaip pat yra reikalinga rezervuoti tam tikras sritis tam tikroms operacijoms vykdyti, ir persiųsti duomenis, reikalingus šioms operacijoms į tokias sritis. Tokių operacijų pavyzdys gali būti tam tikri skaičiavimai. Kitas pavyzdys yra surištas su portų įvedimu/išvedimu, kai porto elgsena gali perdengti tam tikrą sritį.

TREČIOS EILĖS REDUKUOJANTIS PROCESORIUS (TOR)

Išraiškos pernešamos išradime aprašytame antros eilės redukuojančiame procesoriuje. Vienok redukuojantys procesoriai gali ir nebūti tam tikroje geografinėje vietoje. To pačio procesoriaus dalis gali būti pvz. Švedijoje, o kita - Australijoje. Iš to išplaukiant, pateikiamas trečios eilės redukuojantis procesorius, parodytas pieš. 41, įjungiantis kelis antros eilės procesorius. Trys SOP'ai - SOP1, SOP2, SOP3 yra parodyti pieš. 41. SOP1 yra parodytas kaip kvadratinės matricos magistralinis SOP'as.

Asociatyvinis adresavimas prieš fizinį adresavimą

Yra reikalinga sumažinti ryšių skaičių tarp skirtingų geografinių vietų, kadangi didėjant atstumui tarp vietų didėja duomenų perdavimo kaina ir greitis. Asociatyvinė paieška reikalauja per daug komunikavimo ryšių, kadangi kiekvienas sistemos mazgas dalyvauja paieškoje. Šiuo atveju yra prasmė panaudoti fizinį adresavimo būdą, sutaupant ilgų atsumų komunikavimo laiką. Fizinis adresavimas šiuo atveju reiškia, kad mes tiesiogiai adresuojame tam tikrą vietą, t. y. gavėjas jau žinomas iš anksto. Susumuojant, geriausia yra naudoti asociatyvinį adresavimą SOP'o viduje, arba tarp lokaliai

išdėstytų SOP'ų, ir naudoti fizinį adresavimą tarp tokių SOP'ų ar jų grupių. Reikia pažymėti, kad asociatyvinis adresavimas SOP'o viduje yra apribotas srities panaudojimu.

Bet egzistuoja situacija, kai apie sūnaus redukavima turi būti pranešta jo tėvams. Antros eilės procesoriuje tai yra padaroma, panaudojant sritis, kur kiekvienas išraiškų tinklas sutraukiamas įmanomai mažesnę sritį. Trečios eilės procesoriuje prisideda dar viena problema - globalinio adresavimo problema.

Globalinis adresavimas

Objektinės atminties vietos "adresuojamos" pagal savo turinį, o ne pagal fizinį adresą, t. y. yra naudojamas asociatyvinis adresavimas. Šis būdas yra netinkamas globaliniam lygiui, kadangi reikalauja daug ilgų atstumų komunikacinių ryšių. Todėl globalinis adresavimas yra realizuotas labiau tradiciniu būdu. Užuot atlikus globalinę paiešką ir pakeitimą, kai sūnus yra redukuotas, yra atliekama lokalinė asociatyvinė paieška ir pakeitimas, t. y. operacija yra atliekama antros eilės procesoriaus, turinčio tą sūnų, viduje. Toliau įvykdomas, jeigu reikia, globalinis adresavimas, t. y. kai redukuoto sūnaus tėvas yra kitoje geografinėje vietoje. Globalinis adresavimas yra neasociatyvinio pobūdžio, t. y. naudojamas fizinis adresavimas. Fizinis adresas identifikuoja SOP'ą, kuris gali įjungti skirtingus SOP'us. Perduodama informacija įjungia globalinį adresą, skirtą sūninei išraiškai id_g, t. y. identifikatorius id_g yra unikalus visiems procesoriams sistemoje. Informacija taip pat įjungia redukuotą sūnų. Kiekviename geografiškai atskirtame SOP'e yra duomenų struktūra, įjungianti informaciją apie sūnus, turinčius tėvus už SOP'o ribų, kiekvienas sūnus turi nelokalinių tėvų sąrašą. Ši duomenų struktūra žemiau bus vadinama globaline išraiška C_g. Globalinė išraiška C_g gali įjungti ir papildomą informaciją.

Yra keli keliai globalinio pakeitimo operacijoms atlikti, čia globalinis pakeitimas suprantamas, kad mažiausiai vienas tėvas yra kitame SOP'e, negu sūnus.

Globalinis pakeitimas be adreso transliavimo

Kai išraiška yra redukuota, redukuotą išraišką žymintis identifikatorius ieškomas globalinėje išraiškoje C_g. Jeigu joje randamas

identifikatorius, reiškia kad redukuota išraiška turi mažiausiai vieną nelokalinį tėvą SOP'o išorėje - reikalingas globalinis pakeitimas. Globalinis pakeitimas yra nereikalingas, kai identifikatorius nerastas. Abiem atvejais, viduje SOP'o atliekama asocociatyvinė paieška ir pakeitimas. Jeigu reikalingas globalinis pakeitimas yra peržiūrimas nelokalinių tėvų sąrašas globalinėje išraiškoje C_g. Kiekvienam sąraše esančiam tėvui yra pasiunčiamas pranešimas, adresuotas SOP'ui, kuriame yra tas tėvas. Pranešimas įjungia globalinį redukuotos išraiškos adresą ir informaciją apie pačią redukavimo operaciją. Gaunantysis SOP'as įvykdo asociatyvinę paiešką ir pakeitimą, pakeisdamas kiekvieną atitinkantį globalinį identifikatorių redukavimo rezultatu. Protokolas reikalauja, kad kiekvienas identifikatorius būtų unikalus, t. y. identifikatoriaus bitinis piešinys turi būti unikalus visose sistemos procesoriuose.

Šio protokolo realizavime kiekvienas SOP'as turi savo unikalų numerį ir kiekvienas identifikatorius SOP'o numerį laiko užkoduotą savo bitiniame piešinyje, kas reiškia, kad fizinis SOP'o adresas atsispindi ir identifikatoriaus bitiniame piešinyje. Kiekvienas SOP'as "prisiskiria" sau identifikatorių diapazoną, ir kai į sistemą yra įjungiamas naujas SOP'as, jam priskiriamas naujas diapazonas, kurio viduje nėra identifikatorių.

Pavyzdys: SOP1 priklauso identifikatorių diapazonas 100-199, SOP2 priklauso identifikatorių diapazonas 200-299, ..., SOP8 priklauso identifikatorių diapazonas 800-899. Naujas SOP'as, įjungtas į sistemą gauna SOP'o numerį 9 ir jam priskiriamas identifikatorių diapazonas 900-999. Pieš. 42A parodytos pavyzdžio struktūros dalys. Pieš. 42B parodytos trečios eilės procesoriaus, įjungiančio SOP1, SOP6 ir SOP9, dalys. Pieš. 42C parodytos globalinės išraiškos C_g dalys. Sūnus, pažymėtas identifikatoriumi 623, redukuotas. Atliekama lokalinė paieška ir pakeitimas, pakeičiant identifikatorių 623 rezultatu, kurį tas identifikatorius žymi. Globalinėje išraiškoje C_g surandama sūnaus nelokaliniai tėvai, pažymėti 121 ir 999, kaip sąrašo elementai (pieš. 42C). SOP1 ir SOP9 yra pasiunčiami pranešimai su informacija apie redukavimą, t. y. redukuotas identifikatorius, redukavimo rezultatas ir kt. SOP1 ir SOP9 atlieka paiešką ir pakeitimą pakeisdami identifikatorių 623 rezultatu, kurį tas identifikatorius žymi. Globalinės išraiškos C_g duomenų struktūra dabar gali būti modifikuota, pašalinant sūnaus globalinių tėvų

sąrašą, jeigu to reikia.

Vietoj nelokalinių tėvų sąrašo saugojimo, gali būti naudojamas SOP'ų sąrašas su naudojamų tėvų identifikatoriais.

Globalinis pakeitimas su adreso transliavimu

Šiuo atveju išskiriami globaliniai ir lokaliniai adresai, kur lokalinis adresas yra unikalus SOP'o viduje ir globalinis adresas yra unikalus visuose sistemos SOP'uose. Pagrindinis skirtumas tarp šio būdo ir būdo, aprašyto aukščiau yra tas, kad taikant šį pakeitimo būdą reikalingas adresų transliavimas.

Adresų transliavimas gali būti tvarkomas globalinės išraiškos C_g pagalba. Išraiška C_g turi informaciją apie SOP identifikatorių diapazoną. Ji naudoja šią informaciją globalinių adresų transliavimui į lokalinius adresus.

Duomenų perdavimas ir bendravimo protokolas trečios eilės redukuojančiam procesoriui gali būti realizuotas, panaudojant nusistovėjusius duomenų perdavimo principus.

Galimas ir aukštesnės eilės procesorių sukūrimas, panaudojant įprastas technologijas kartu su aprašytomis aukščiau, naudojant pirmos, antros ir/arba trečios eilės procesorius kaip sudėtines dalis.

Nežiūrint į tai, kad išradimas aprašytas su nuorodomis į tam tikrą įranga, galimi įvairūs pakeitimai ir elementų patobulinimai, nepažeidžiant išradimo principų ir ideologijos. Galimos ir įvairios modifikacijos, nenutolstant nuo išradimo esmės.

700

PRIEDAS 1

CENTRINIO VALDYMO ĮRENGINIO 6 OPERACIJOS PAVYZDYS

Operacija: unify_structure_full, 2 naudojami sūnūs su 3 elementais kiekviename
H-išraiška, kur "x" - bet kas, išskyrus "nenaudojama"

(par (x x x) par (x x x))

Bazinės ląstelės turinys, išreikštas raktiniais žodžiais

Pos	Attribute	Head	Num
ID:	clos exec node val unify	indirect cls	x
ENV:		indirect	x
F0:	simple unused	unused	
F1:	simple unused	unused	
F2:	simple unused	unused	
F3:	simple unused	unused	
S0,0:	clos idle node val par	x	x
S0,1:		x	x
S0,2:		x	x
S0,3:		unused	
S1,0:	clos idle node val par	x	x
S1,1:		x	x
S1,2:		x	x
S1,3:		unused	
S2,0:	simple unused	unused	
S2,1:		unused	
S2,2:		unused	
S2,3:		unused	
S3,0:	simple unused	unused	
S3,1:		unused	
S3,2:		unused	
S3,3:		unused	

Bazinės ląstelės turinys, išreikštas kodais

Pos	Attribute	Head	Num
ID:	1 11 0 0 0111	101010	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ENV:		1xxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
F0:	0 00 0 0 0000	000000	00000000000000000000000000000000
F1:	0 00 0 0 0000	000000	00000000000000000000000000000000
F2:	0 00 0 0 0000	000000	00000000000000000000000000000000
F3:	0 00 0 0 0000	000000	00000000000000000000000000000000
S0,0:	1 00 0 0 0001	xxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
S0,1:		xxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
S0,2:		xxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
S0,3:		000000	00000000000000000000000000000000
S1,0:	1 00 0 0 0001	xxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
S1,1:		xxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
S1,2:		xxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
S1,3:		000000	00000000000000000000000000000000
S2,0:	0 00 0 0 0000	000000	00000000000000000000000000000000
S2,1:		000000	00000000000000000000000000000000
S2,3:		000000	00000000000000000000000000000000
S2,3:		000000	00000000000000000000000000000000
S3,0:	0 00 0 0 0000	000000	00000000000000000000000000000000
S3,1:		000000	00000000000000000000000000000000
S3,3:		000000	00000000000000000000000000000000
S3,3:		000000	00000000000000000000000000000000

Operacija: unify_sructure_full, naudojami sūnūs su 3 elementais kiekviename Fazė a

Pos	Plane	swa1	swb1	swQ	swa0	swb0	swN	swE	swVO	swVI	swC	swHO	swHI	swD	swCON
ID	Tcs	off	---	off	on	---	---	---	off	off	off	---	---	---	on
ID	Tl	off	---	off	on	---	---	---	off	off	off	---	---	---	on
ID	Tw	off	---	off	on	---	---	---	off	off	off	---	---	---	on
ID	Tt	off	---	off	on	---	---	---	off	off	off	---	---	---	on
ID	Eh	off	---	off	on	---	---	---	off	off	off	---	---	---	---
ID	En	off	---	off	on	---	---	---	off	off	off	---	---	---	---
ENV	Eh	off	---	off	on	---	---	---	off	off	---	---	---	---	---
ENV	En	off	---	off	on	---	---	---	off	off	---	---	---	---	---
F0	Tcs	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F0	Tl	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F0	Tw	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F0	Tt	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F0	Eh	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F0	En	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F1	Tcs	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F1	Tl	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F1	Tw	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F1	Tt	off	---	off	on.	---	---	---	off	off	off	off	off	---	off
F1	Eh	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F1	En	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F2	Tcs	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F2	Tl	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F2	Tw	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F2	Tt	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F2	Eh	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F2	En	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F3	Tcs	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F3	Tl	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F3	Tw	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F3	Tt	off	---	off	on	---	---	---	off	off	off	off	off	---	off
F3	Eh	off	---	off	on	---	---	---	off	off	off	off	off	---	---
F3	En	off	---	off	on	---	---	---	off	off	off	off	off	---	---

Pos	Plane	swa1	swb1	swQ	swa0	swb0	swN	swE	swVO	swVI	swC	swHO	swHI	swD	swCON
S0,0	Tcs	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S0,0	Tl	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S0,0	Tw	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S0,0	Tt	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S0,0	Eh	off	off	off	on	off	off	off	off	off	---	off	off	---	---
S0,0	En	off	on	off	on	off	off	off	off	off	---	off	off	---	---
S0,1	Eh	off	on	off	on	off	off	off	off	off	---	off	off	---	---
S0,1	En	off	on	off	on	off	off	off	off	off	---	off	off	---	---
S0,2	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S0,2	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S0,3	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S0,3	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S1,0	Tcs	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S1,0	Tl	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S1,0	Tw	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S1,0	Tt	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S1,0	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S1,0	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S1,1	Eh	off	off	off	on	off	off	off	off	off	---	off	off	off	---
S1,1	En	off	off	off	on	off	off	off	off	off	---	off	off	off	---
S1,2	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S1,2	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S1,3	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S1,3	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S2,0	Tcs	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S2,0	Tl	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S2,0	Tw	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S2,0	Tt	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S2,0	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S2,0	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S2,1	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S2,1	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S2,2	Eh	off	off	off	on	off	off	off	off	off	---	off	off	on	---
S2,2	En	off	off	off	on	off	off	off	off	off	---	off	off	on	---
S2,3	Eh	off	on	off	on	off	off	off	off	off	---	off	off	---	---
S2,3	En	off	on	off	on	off	off	off	off	off	---	off	off	---	---
S3,0	Tcs	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S3,0	Tl	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S3,0	Tw	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S3,0	Tt	off	off	off	on	off	---	off	off	off	---	off	off	---	---
S3,0	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S3,0	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S3,1	Eh	off	on	off	on	off	off	off	off	off	---	off	off	off	---
S3,1	En	off	on	off	on	off	off	off	off	off	---	off	off	off	---
S3,2	Eh	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S3,2	En	off	on	off	on	off	off	off	off	off	---	off	off	on	---
S3,3	Eh	off	on	off	on	off	off	off	off	off	---	off	off	---	---
S3,3	En	off	on	off	on	off	off	off	off	off	---	off	off	---	---

Pos	Plane	State
SWv0,cv	Tcs	off
SWv0,cv	Tl	off
SWv0,cv	Tw	off
SWv0,cv	Tt	off
SWv0,cv	Eh	off
SWv0,cv	En	off
SWv1,cv	Tcs	off
SWv1,cv	Tl	off
SWv1,cv	Tw	off
SWv1,cv	Tt	off
SWv1,cv	Eh	off
SWv1,cv	En	off
SWv2,cv	Tcs	off
SWv2,cv	Tl	off
SWv2,cv	Tw	off
SWv2,cv	Tt	off
SWv2,cv	Eh	off
SWv2,cv	En	off
SWv3,cv	Tcs	off
SWv3,cv	Tl	off
SWv3,cv	Tw	off
SWv3,cv	Tt	off
SWv3,cv	Eh	off
SWv3,cv	En	off
SWch,h0	Tcs	off
SWch,h0	Tl	off
SWch,h0	Tw	off
SWch,h0	Tt	off
SWch,h0	Eh	off
SWch,h0	En	off
SWid,h0	Tcs	off
SWid,h0	Tl	off
SWid,h0	Tw	off
SWid,h0	Tt	off
SWid,h0	Eh	off
SWid,h0	En	off
SWenv,h0	Eh	off
SWenv,h0	En	off

Pos	Plane	State
SWv0	Tcs	off
SWv0	Tl	off
SWv0	Tw	off
SWv0	Tt	off
SWv0	Eh	off
SWv0	En	off
SWv1,h0	Tcs	off
SWv1,h0	Tl	off
SWv1,h0	Tw	off
SWv1,h0	Tt	off
SWv1,h0	Eh	off
SWv1,h0	En	off
SWv2,h0	Tcs	off
SWv2,h0	Tl	off
SWv2,h0	Tw	off
SWv2,h0	Tt	off
SWv2,h0	Eh	off
SWv2,h0	En	off
SWv3,h0	Tcs	off
SWv3,h0	Tl	off
SWv3,h0	Tw	off
SWv3,h0	Tt	off
SWv3,h0	Eh	off
SWv3,h0	En	off
SWv1	Tcs	off
SWv1	Tl	off
SWv1	Tw	off
SWv1	Tt	off
SWv1	Eh	off
SWv1	En	off
SWv2	Tcs	off
SWv2	Tl	off
SWv2	Tw	off
SWv2	Tt	off
SWv2	Eh	off
SWv2	En	off
SWv3	Tcs	off
SWv3	Tl	off
SWv3	Tw	off
SWv3	Tt	off
SWv3	Eh	off
SWv3	En	off

Operacija: unify_structure_full, 2 naudojami sūnūs su 3 elementais kiekviename

Fazė b

Pos	Plane	swa1	swb1	swQ	swa0	swb0	swN	swE	swVO	swVI	swC	swHO	swHI	swD	swCON
ID	Tcs	on	---	off	off	---	---	---	off	on	off	---	---	---	on
ID	Tl	on	---	off	off	---	---	---	off	off	on	---	---	---	on
ID	Tw	on	---	off	off	---	---	---	off	off	on	---	---	---	on
ID	Tt	on	---	off	off	---	---	---	off	on	off	---	---	---	on
ID	Eh	off	---	off	off	---	---	---	off	on	off	---	---	---	---
ID	En	off	---	off	off	---	---	---	off	on	off	---	---	---	---
ENV	Eh	off	---	off	off	---	---	---	off	off	---	---	---	---	---
ENV	En	off	---	off	off	---	---	---	off	off	---	---	---	---	---
F0	Tcs	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F0	Tl	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F0	Tw	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F0	Tt	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F0	Eh	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F0	En	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F1	Tcs	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F1	Tl	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F1	Tw	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F1	Tt	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F1	Eh	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F1	En	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F2	Tcs	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F2	Tl	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F2	Tw	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F2	Tt	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F2	Eh	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F2	En	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F3	Tcs	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F3	Tl	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F3	Tw	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F3	Tt	off	---	off	off	---	---	---	off	off	off	off	off	---	off
F3	Eh	off	---	off	off	---	---	---	off	off	off	off	off	---	---
F3	En	off	---	off	off	---	---	---	off	off	off	off	off	---	---

Pos	Plane	swa1	swb1	swQ	swa0	swb0	swN	swE	swVO	swVI	swC	swHO	swHI	swD	swCON
S0,0	Tcs	on	off	off	off	on	---	off	off	on	---	on	off	---	---
S0,0	Tl	on	off	off	off	on	---	off	off	on	---	on	off	---	---
S0,0	Tw	on	off	off	off	on	---	off	off	on	---	on	off	---	---
S0,0	Tt	on	off	off	off	on	---	off	off	on	---	on	off	---	---
S0,0	Eh	off	off	off	off	on	off	off	off	off	---	on	off	---	---
S0,0	En	off	off	off	off	on	off	off	off	off	---	on	off	---	---
S0,1	Eh	on	off	off	off	on	off	off	off	off	---	on	off	---	---
S0,1	En	on	off	off	off	on	off	off	off	off	---	on	off	---	---
S0,2	Eh	off	off	off	off	on	off	off	off	off	---	on	off	on	---
S0,2	En	off	off	off	off	on	off	off	off	off	---	on	off	on	---
S0,3	Eh	off	off	off	off	on	off	off	off	off	---	on	off	on	---
S0,3	En	off	off	off	off	on	off	off	off	off	---	on	off	on	---
S1,0	Tcs	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S1,0	Tl	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S1,0	Tw	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S1,0	Tt	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S1,0	Eh	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S1,0	En	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S1,1	Eh	off	off	off	off	on	off	off	off	off	---	off	off	off	---
S1,1	En	off	off	off	off	on	off	off	off	off	---	off	off	off	---
S1,2	Eh	on	off	off	off	on	off	off	off	off	---	off	off	on	---
S1,2	En	on	off	off	off	on	off	off	off	off	---	off	off	on	---
S1,3	Eh	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S1,3	En	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,0	Tcs	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S2,0	Tl	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S2,0	Tw	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S2,0	Tt	on	off	off	off	on	---	off	off	on	---	off	off	---	---
S2,0	Eh	on	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,0	En	on	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,1	Eh	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,1	En	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,2	Eh	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,2	En	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S2,3	Eh	on	off	off	off	on	off	off	off	off	---	off	off	---	---
S2,3	En	on	off	off	off	on	off	off	off	off	---	off	off	---	---
S3,0	Tcs	off	off	off	off	on	---	off	off	off	---	off	off	---	---
S3,0	Tl	off	off	off	off	on	---	off	off	off	---	off	off	---	---
S3,0	Tw	off	off	off	off	on	---	off	off	off	---	off	off	---	---
S3,0	Tt	off	off	off	off	on	---	off	off	off	---	off	off	---	---
S3,0	Eh	on	off	off	off	on	off	off	off	off	---	off	off	on	---
S3,0	En	on	off	off	off	on	off	off	off	off	---	off	off	on	---
S3,1	Eh	on	off	off	off	on	off	off	off	off	---	off	off	off	---
S3,1	En	on	off	off	off	on	off	off	off	off	---	off	off	off	---
S3,2	Eh	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S3,2	En	off	off	off	off	on	off	off	off	off	---	off	off	on	---
S3,3	Eh	off	off	off	off	on	off	off	off	off	---	off	off	---	---
S3,3	En	off	off	off	off	on	off	off	off	off	---	off	off	---	---

Pos	Plane	State
SWv0,cv	Tcs	on
SWv0,cv	Tl	on
SWv0,cv	Tw	on
SWv0,cv	Tt	on
SWv0,cv	Eh	off
SWv0,cv	En	off
SWv1,cv	Tcs	on
SWv1,cv	Tl	on
SWv1,cv	Tw	on
SWv1,cv	Tt	on
SWv1,cv	Eh	off
SWv1,cv	En	off
SWv2,cv	Tcs	on
SWv2,cv	Tl	on
SWv2,cv	Tw	on
SWv2,cv	Tt	on
SWv2,cv	Eh	off
SWv2,cv	En	off
SWv3,cv	Tcs	on
SWv3,cv	Tl	on
SWv3,cv	Tw	on
SWv3,cv	Tt	on
SWv3,cv	Eh	off
SWv3,cv	En	off
SWch,h0	Tcs	off
SWch,h0	Tl	off
SWch,h0	Tw	off
SWch,h0	Tt	off
SWch,h0	Eh	off
SWch,h0	En	off
SWid,h0	Tcs	on
SWid,h0	Tl	on
SWid,h0	Tw	on
SWid,h0	Tt	on
SWid,h0	Eh	off
SWid,h0	En	off
SWenv,h0	Eh	off
SWenv,h0	En	off

Pos	Plane	State
SWv0	Tcs	off
SWv0	Tl	off
SWv0	Tw	off
SWv0	Tt	off
SWv0	Eh	off
SWv0	En	off
SWv1,h0	Tcs	off
SWv1,h0	Tl	off
SWv1,h0	Tw	off
SWv1,h0	Tt	off
SWv1,h0	Eh	off
SWv1,h0	En	off
SWv2,h0	Tcs	off
SWv2,h0	Tl	off
SWv2,h0	Tw	off
SWv2,h0	Tt	off
SWv2,h0	Eh	off
SWv2,h0	En	off
SWv3,h0	Tcs	off
SWv3,h0	Tl	off
SWv3,h0	Tw	off
SWv3,h0	Tt	off
SWv3,h0	Eh	off
SWv3,h0	En	off
SWv1	Tcs	off
SWv1	Tl	off
SWv1	Tw	off
SWv1	Tt	off
SWv1	Eh	off
SWv1	En	off
SWv2	Tcs	off
SWv2	Tl	off
SWv2	Tw	off
SWv2	Tt	off
SWv2	Eh	off
SWv2	En	off
SWv3	Tcs	off
SWv3	Tl	off
SWv3	Tw	off
SWv3	Tt	off
SWv3	Eh	off
SWv3	En	off

Operaoija: unify_structure_full, 2 naudojamų sūnūs su 3 elementais kiekviename Fazė 0

Pos	Plane	swa1	swb1	swQ	swa0	swb0	swN	swE	swVO	swVI	swC	swHO	swHI	swD	swCON
ID	Tcs	on	---	on	on	---	---	---	off	off	off	---	---	---	on
ID	Tl	on	---	on	on	---	---	---	off	off	off	---	---	---	on
ID	Tw	on	---	on	on	---	---	---	off	off	off	---	---	---	on
ID	Tt	on	---	on	on	---	---	---	off	off	off	---	---	---	on
ID	Eh	on	---	on	on	---	---	---	off	off	off	---	---	---	---
ID	En	on	---	on	on	---	---	---	off	off	off	---	---	---	---
ENV	Eh	on	---	on	on	---	---	---	off	off	---	---	---	---	---
ENV	En	on	---	on	on	---	---	---	off	off	---	---	---	---	---
F0	Tcs	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F0	Tl	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F0	Tw	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F0	Tt	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F0	Eh	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F0	En	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F1	Tcs	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F1	Tl	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F1	Tw	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F1	Tt	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F1	Eh	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F1	En	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F2	Tcs	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F2	Tl	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F2	Tw	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F2	Tt	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F2	Eh	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F2	En	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F3	Tcs	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F3	Tl	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F3	Tw	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F3	Tt	on	---	on	on	---	---	---	off	off	off	off	off	---	off
F3	Eh	on	---	on	on	---	---	---	off	off	off	off	off	---	---
F3	En	on	---	on	on	---	---	---	off	off	off	off	off	---	---

Pos	Plane	swa1	swb1	swQ	swa0	swb0	swN	swE	swVO	swVI	swC	swHO	swHI	swD	swCON
S0,0	Tcs	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S0,0	Tl	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S0,0	Tw	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S0,0	Tt	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S0,0	Eh	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S0,0	En	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S0,1	Eh	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S0,1	En	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S0,2	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S0,2	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S0,3	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S0,3	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,0	Tcs	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S1,0	Tl	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S1,0	Tw	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S1,0	Tt	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S1,0	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,0	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,1	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,1	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,2	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,2	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,3	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S1,3	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,0	Tcs	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S2,0	Tl	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S2,0	Tw	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S2,0	Tt	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S2,0	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,0	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,1	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,1	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,2	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,2	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S2,3	Eh	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S2,3	En	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S3,0	Tcs	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S3,0	Tl	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S3,0	Tw	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S3,0	Tt	on	on	on	on	on	---	off	off	off	---	off	off	---	---
S3,0	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S3,0	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S3,1	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S3,1	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S3,2	Eh	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S3,2	En	on	on	on	on	on	off	off	off	off	---	off	off	off	---
S3,3	Eh	on	on	on	on	on	off	off	off	off	---	off	off	---	---
S3,3	En	on	on	on	on	on	off	off	off	off	---	off	off	---	---

Pos	Plane	State
SWv0,cv	Tcs	off
SWv0,cv	Tl	off
SWv0,cv	Tw	off
SWv0,cv	Tt	off
SWv0,cv	Eh	off
SWv0,cv	En	off
SWv1,cv	Tcs	off
SWv1,cv	Tl	off
SWv1,cv	Tw	off
SWv1,cv	Tt	off
SWv1,cv	Eh	off
SWv1,cv	En	off
SWv2,cv	Tcs	off
SWv2,cv	Tl	off
SWv2,cv	Tw	off
SWv2,cv	Tt	off
SWv2,cv	Eh	off
SWv2,cv	En	off
SWv3,cv	Tcs	off
SWv3,cv	Tl	off
SWv3,cv	Tw	off
SWv3,cv	Tt	off
SWv3,cv	Eh	off
SWv3,cv	En	off
SWch,h0	Tcs	off
SWch,h0	Tl	off
SWch,h0	Tw	off
SWch,h0	Tt	off
SWch,h0	Eh	off
SWch,h0	En	off
SWid,h0	Tcs	off
SWid,h0	Tl	off
SWid,h0	Tw	off
SWid,h0	Tt	off
SWid,h0	Eh	off
SWid,h0	En	off
SWenv,h0	Eh	off
SWenv,h0	En	off

Pos	Plane	State
SWv0	Tcs	off
SWv0	Tl	off
SWv0	Tw	off
SWv0	Tt	off
SWv0	Eh	off
SWv0	En	off
SWv1,h0	Tcs	off
SWv1,h0	Tl	off
SWv1,h0	Tw	off
SWv1,h0	Tt	off
SWv1,h0	Eh	off
SWv1,h0	En	off
SWv2,h0	Tcs	off
SWv2,h0	Tl	off
SWv2,h0	Tw	off
SWv2,h0	Tt	off
SWv2,h0	Eh	off
SWv2,h0	En	off
SWv3,h0	Tcs	off
SWv3,h0	Tl	off
SWv3,h0	Tw	off
SWv3,h0	Tt	off
SWv3,h0	Eh	off
SWv3,h0	En	off
SWv1	Tcs	off
SWv1	Tl	off
SWv1	Tw	off
SWv1	Tt	off
SWv1	Eh	off
SWv1	En	off
SWv2	Tcs	off
SWv2	Tl	off
SWv2	Tw	off
SWv2	Tt	off
SWv2	Eh	off
SWv2	En	off
SWv3	Tcs	off
SWv3	Tl	off
SWv3	Tw	off
SWv3	Tt	off
SWv3	Eh	off
SWv3	En	off

Plane abbreviations

Abbr.	Name	Number of bits
Tcs	clos/simple	1
Tl	lazy	2
Tw	where	1
Tt	type (form+tag)	5
Eh	head	6
En	num	32

PRIEDAS 2

Kontrolinis uždavinys 1

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (1, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (1, X, X, X)
sign2           (X, X, X, X)
sign3           (X, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (1, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (X, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 0; 0; 0; 0; 0; 1]
```

MIKROINSTRUKCIJŲ LINIJOS

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 1, 0, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 2

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (0, X, X, X)
sign2           (0, X, X, X)
sign3           (X, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (0, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (0, X, X, X)
sign2           (X, X, X, X)
sign3           (X, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 3

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
```

```
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (0, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (X, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 4

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (0, X), (X, X))
insig2          ((X, X), (0, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (X, X, X, X)
sign2           (X, X, X, X)
sign3           (X, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]
```

```
c2
c2_1            (0, 0, 0, 0, 1, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

915

kontrolinis uždavinys 3

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (1, X), (X, X))
sign1           (X, X, X, X)
sign2           (X, X, X, X)
sign3           (1, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (1, X), (X, X))
sign1           (X, X, X, X)
sign2           (X, X, X, X)
sign3           (1, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 1; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 1, 0, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
```

```
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 1, 0, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 4

---

būsena, reiškiančios linijos

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (X, X, X, X)
sign3           (1, X, X, 0)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 1, 0, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 0, 1, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (0, 1); (1, 0); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 5

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (1, X, X, 1)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, 1, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos-sąlyga2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 1), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (0, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 1, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 3

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
```

```
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 0), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (0, X, X, 0)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 1, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 4

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (1, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 1; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 5

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (0, X, X, X)
```

```
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 1; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [1; 1; 1; 1; 1]

c2
c2_1            (0, 0, 0, 1, 0, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 6

būsena, reiškiančios linijos - sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (1, X, X, 1)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, 0, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos - sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 1), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (0, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 0, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 3

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
```

```
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 0), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (0, X, X, 0)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 0, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 4

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (1, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 1; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 5

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, X), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (0, X, X, X)
```

```
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 0; 1; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 0]

c2
c2_1            (0, 0, 0, 1, 0, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 7

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (1, X, X, 1)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, 1, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (0, X), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (1, X, X, 1)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, 0, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
```

```
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 0, 1, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (0, 1); (1, 0); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 8

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 1), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (0, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 0, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 1), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (0, X, X, X)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 1, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
```

```
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(0, 1); (0, 1); (0, 1); (1, 0); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 1, 0, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [0; 0])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 9

būsena, reiškiančios linijos

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 0), (X, X))
sign1           (X, X, X, X)
sign2           (X, X, X, X)
sign3           (0, X, X, 1)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, X, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 1, 0, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 0, 1, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (0, 1); (1, 0); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [1; 1])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

kontrolinis uždavinys 10

---

būsena, reiškiančios linijos- sąlyga 1

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 0), (X, X))
sign1           (X, X, X, X)
sign2           (0, X, X, X)
sign3           (0, X, X, 0)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 0, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

būsena, reiškiančios linijos- sąlyga 2

```
cmp_prec        X
cmp01           (X, X)
cmp12           (X, X)
cmp23           (X, X)
cmp_exp12       (X, X)
insig1          ((X, X), (X, X), (X, X))
insig2          ((X, X), (X, X), (X, X))
insig3          ((X, X), (X, 0), (X, X))
sign1           (X, X, X, X)
sign2           (1, X, X, X)
sign3           (0, X, X, 0)
gr_i1           (X, X, X)
c_ei1           X
gr_d1           (X, X, X)
c_ed1           X
co_limit1       (X, X)
gr_i2           (X, X, X)
c_ei2           X
gr_d2           (X, X, X)
c_ed2           X
co_limit2       (X, X)
gr_a1           (X, X, X)
gr_a2           (X, 1, X)
gr_a3           (X, X, X)

ins             [1; 1; 0; 0; 0; 0; 0; 0; 1]
```

mikroinstrukcijų linijos

```
c1
c1_1            (0, 0, 0, 0, 1, 0)
c1_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
```

```
c1_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c1_4            [(0, 1); (0, 1); (1, 0); (0, 1); (0, 1); (0, 1)]
c1_5            [(0, 1); (1, 0); (0, 1)]
c1_6            [(0, 1); (0, 1); (0, 1); (1, 0); (0, 1)]
c1_7            (0, [(0, 1); (1, 0)], [0; 0])
c1_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c1_10           [0; 0; 0; 0; 1]

c2
c2_1            (0, 0, 0, 0, 1, 0)
c2_2            [(0, 1); (0, 1); (1, 0); (0, 1)]
c2_3            [(0, 1); (0, 1); (0, 1); (0, 1); (1, 0)]
c2_4            [(0, 1); (1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_5            [(0, 1); (1, 0); (0, 1)]
c2_6            [(1, 0); (0, 1); (0, 1); (0, 1); (0, 1)]
c2_7            (0, [(0, 1); (1, 0)], [0; 0])
c2_8            (0, [(0, 1); (1, 0)], [0; 0; 0; 0])
c2_10           [0; 0; 0; 0; 1]

c3
c3_1            (1, 0)
c3_2            [(0, 1); (0, 1); (0, 1); (1, 0); (0, 1); (0, 1)]
c3_3            (0, [(0, 1); (1, 0)], [1; 1])
c3_4            [0; 0; 0; 0; 1]
c3_5            ([(1, 0); (0, 1)], 0)

c4              (0, 1)

c5              (0, 0, 0)
```

Apibrėžtis įjungia

1. Redukuojantį procesorių, valdomą programos, turinčią tam tikrą struktūrą, ir pritaikytą redukuoti minėtą struktūrą per tam tikrą redukavimo žingsnių skaičių, pritaikant skirtingus redukavimo būdus, apibūdinamą tuo, šio tipo pirmos eilės procesorius įjungia asociatyvinę, aktyvią atmintį (1, 2) įjungiančią

a. tam tikrą skaičių aktyvių atminties ląstelių (10, 2), kurių kiekviena įjungia informaciją redukavimo operacijų vykdymui, ir

b. ryšio tinklą (t_1, t_2, id, env, v_0, v_1, v_2, v_3, 12, 13, 14, 6, 7, 11, 16, 17) prijungtą prie minėtų atminties ląstelių, pritaikytų asociatyvinei paieškai vykdyti, po kiekvieno redukavimo rezultatas pateikiamas kaip minėtos redukavimo operacijos rezultatas visoms veikiančioms minėtos atminties ląstelėms, įjungiančioms nuorodą į minėtą redukavimą, ir redukavimo rezultatas perduodamas asociatyviniu būdu minėtoms veikiančioms atminties ląstelėms.

2. Redukuojantį procesorių pagal Apibrėžtį 1, apibūdinamą tuo, kad minėtas ryšio tinklas įjungia magistralių posistemę turinčią valdymo linijas (linijas į ir iš 6) ir duomenų linijas (t_1, t_2, id, env, v_0, v_1, v_2, v_3), visos linijos yra prijungtos prie kiekvienos minėtos atminties ląstelės, ir prie valdymo priemonių (6) bendrų visoms minėtoms atminties ląstelėms.

3. Redukuojantį procesorių pagal Apibrėžtį 2 arba 3, apibūdinamą tuo, kad kiekviena atminties ląstelė (2; 10; pieš. 4A, pieš. 4E), kai ji naudojama, turi visą informaciją, reikalingą redukavimo operacijai vykdyti.

4. Redukuojantį procesorių pagal Apibrėžtį 3, apibūdinamą tuo, kad minėta redukavimo informacija taip pat įjungia nuorodą (VALUE/DES) į mažiausiai vieną kitą minėtą atminties ląstelę, atminties ląstelių turinys surišamas minėtos nuorodos(ų) pagalba į medžio struktūrą (pieš 6B, 6G, 7B, 8B; pieš. nuo 5A iki 5F).

5. Redukuojantį procesorių pagal Apibrėžtis 1 - 4, apibūdinamą tuo, kad mažiausiai viena iš minėtų atminties

ląstelių, vadinama bazine ląstele (2), yra pritaikyta vykdyti visus redukavimo būdus, ir likusios minėtos ląstelės, vadinamos objektinės atminties ląstelėmis (10; pieš. 4A), yra pritaikytos tik kai kurių redukavimo operacijų tam tikroms dalims vykdyti.

6. Redukuojantį procesorių pagal Apibrėžtį 5, a p i b ū d i n a m ą tuo, kad minėtos objektinės atminties ląstelės yra įjungtos į asociatyvinę atmintį, turinčią pirmąją magistralės posistemę (any_type, Vr, cpb, set.s, match, r/w.s, r/w.b, r/w.r, Wand.a, Wand.b, Wor, s.a, reset.b, mode.a, mode.a*, prech, ba, mode.b, grant.b, prio ir kt.) skirtą išoriniam valdymui, ir antrąją atminties magistralės posistemę (t1, t2, id, env, v0, v1, v2, v3) skirtą duomenims ir įjungiančią:

keletą minėtų objektinės atminties ląstelių (10) sudėtinės informacijos saugojimui,

priemones (16, 17) mažiausiai vienai žymei minėtose objektinės atminties ląstelėse saugoti, minėta žymė indikuoja mažiausiai minėtos objektinės atminties ląstelės išrinkimo ar neišrinkimo būseną,

priemones paieškos operacijai vykdyti minėtose objektinės atminties ląstelėse minėtos žymės nustatymui, ir

prioriteto dekoderį (11) prijungtą prie visų minėtų objektinės atminties ląstelių ir kuris išrenka vieną iš kelių minėtų objektinės atminties ląstelių.

7. Redukuojantį procesorių pagal Apibrėžtį 5, a p i b ū d i n a m ą tuo, kad mažiausiai viena bendra magistralė (12, 13, 14) yra pateikta loginių operacijų, IR ir ARBA tipo, tarp minėtų atminties ląstelių vykdymui, ir yra priemonės (17) kiekvienoje atminties ląstelėje bendravimui su minėtomis magistralėmis ir valdant minėtos atminties ląstelės veiklą vykstančiose loginėse operacijose.

8. Redukuojantį procesorių pagal Apibrėžtį 6 arba 7, a p i b ū d i n a m ą tuo, kad kiekviena objektinės atminties ląstelė įjungia tam tikrą skaičių objektinės atminties duomenų laukų (IDENTIFIER, ENVIROMENT, VALUE/DES.$_0$, VALUE/DES.$_1$, VALUE/DES.$_2$, VALUE/DES.$_3$), kiekvienas objektinės atminties duomenų laukas gali saugoti duomenų žodį, vadinamą skaitmeniniu žodžiu, ir požymius,

esančius požymių žodžio formoje.

9. Redukuojantį procesorių pagal Apibrėžtis 6 - 8, apibūdinamą tuo, kad kiekviena atminties ląstelė įjungia mažiausiai vieną atributų saugojimo lauką (LAZY, WHERE, TYPE), parodantį minėtos atminties ląstelės turinio būseną ar būsenas.

10. Redukuojantį procesorių pagal Apibrėžtis 5 - 10, apibūdinamą tuo, kad minėta bazinė ląstelė yra aritmetinis įrenginys, skirtas struktūriniam aritmetiniam skaičiavimui, ir įjungiantis:

a) mažiausiai vieną įvedimo/išvedimo priemonę (v0, v1, v2, v3, id, env) duomenų sąrašų įvedimui ir išvedimui į ir iš minėtos objektinės atminties ląstelės,

b) kelis registrus ($S_{0,0}$ iki $S_{3,3}$, F0 iki F3, ID, ENV) pritaikytus duomenų žodžio saugojimui, kur kiekvienas duomenų žodis turi požymių dalį, požymių žodį ir informacinę dalį, skaitmeninį žodį, minėta požymių dalis įjungia požymį, nurodantį registro naudojimą, kiekvienas iš minėtų sąrašų yra saugomas nustatytuose minėtuose registruose, minėto registro minėta požymių dalis, pažymėta kaip naudojama todėl, kad minėtis sąrašai saugo mažiausiai vieną savo dalį šiame registre, ir minėti sąrašai, saugantys savo dalį minėtame veikiančiame registre, įjungia sąrašo instrukciją, nurodančią kokio tai tipo sąrašas, ir ryšiai tarp minėtų sąrašų realizuojami sutvarkant minėtus sąrašus minėtuose registruose,

c) valdymo priemones (6) minėtų registrų valdymui, minėtuose registruose saugomų sąrašų instrukcijų naudojimui, pertvarkant minėtus sąrašus minėtuose registruose, ir įvedimui/išvedimui registrų turinio pagal minėtų sąrašų instrukcijas.

11. Redukuojantį procesorių pagal Apibrėžtį 10, apibūdinamą tuo, kad saugomi minėtuose registruose minėti sąrašai yra sutvarkyti kaip sąrašų medis, kur vienas sąrašas yra šaknies sąrašas.

12. Redukuojantį procesorių pagal Apibrėžtį 11, apibūdinamą tuo, kad yra mažiausiai vienas papildomas registras (ID), kuriame gali būti saugomas saugomo sąrašų medžio identifikatorius.

13. Redukuojantį procesorių pagal Apibrėžtį 11 arba 12, apibūdinamą tuo, kad yra mažiausiai vienas papildomas re-

gistras (ENV), kuriame gali būti saugomas saugomo sąrašų medžio kontekstas.

14. Redukuojantį procesorių pagal Apibrėžtis 10 - 13, įjungiantį registrų ($S_{0,0}$ iki $S_{3,3}$) matricą, turinčią periferinę eilutę ($S_{0,0}$ iki $S_{3,0}$) sudarančią pagrindinius registrus, minėtos matricos stulpeliai sudaro bazinius registrus.

15. Redukuojantį procesorių pagal Apibrėžtį 14, apibūdinamą papildomų registrų (F0 iki F3) įjungimu už minėtos matricos ribų.

16. Redukuojantį procesorių pagal Apibrėžtis 10 - 15, apibūdinamą tuo, kad minėto sąrašų medžio minėtas šaknies sąrašas yra pritaikytas patalpinti skirtinguose registruose, priklausomai nuo numatyto saugoti medžio lygio.

17. Redukuojantį procesorių pagal Apibrėžtis 10 - 16, apibūdinamą tuo, kad informacija, liečianti redukavimo rūšį gali būti gauta iš minėto šaknies sąrašo tipo, ir kur minėtas tipas, jeigu jis neatspindi funkcijos, įjungia instrukcijos kodą, atspindintį vykdomą instrukciją, o jeigu minėtas tipas atspindi funkciją, pirmasis minėto šaknies sąrašo elementas įjungia instrukcijos kodą arba sąrašų medžio šaknis atspindi funkcijos apibrėžimą, minėtos valdymo priemonės (6) yra pritaikytos gauti minėtą informaciją iš minėto šaknies sąrašo.

18. Redukuojantį procesorių pagal Apibrėžtis 10 - 17, apibūdinamą tuo, kad minėti registrai yra sudalinti į logines dalis, atitinkančias bazinės ląstelės plokštumas, kiekviena plokštuma įjungia daugiausiai kiekvieno registro vieną ląstelę, ir kiekviena registro ląstelė gali saugoti vieną informacijos bitą ir kur registrų ląstelės, esančios vienoje plokštumoje, yra sujungiamos tarpusavyje.

19. Redukuojantį procesorių pagal Apibrėžtis 5 - 18, apibūdinamą tuo, kad minėtas ryšio tinklas yra taip pat pritaikytas perduoti informaciją iš objektinės atminties ląstelės į minėtos bazinės ląstelės (2) registrus (ID, ENV, $S_{0,0}$ iki $S_{3,3}$, F0 iki F3) išrenkamus minėto ryšio tinklo (6) pagalba, ir informaciją iš minėtų registrų į vieną ar kelias minėtas objektinės atminties ląsteles, išrainktas minėto tinklo pagalba.

20. Redukuojantį procesorių pagal Apibrėžtis 5 - 19,

apibūdinamą tuo, kad minėtas ryšio tinklas (6) yra pritaikytas perduoti informaciją tarp minėtos bazinės ląstelės registrų ir tarp minėtos objektinės atminties vienos ląstelės, perkeliant jos turinį į minėtus registrus, ir kitos minėtos atminties ląstelės.

21. Redukuojantį procesorių pagal Apibrėžtis 3 - 20, apibūdinamą tuo, kad viena ar kelios minėtos atminties ląstelės (2, 10) yra pritaikytos saugojimui išraiškos, turinčios vykdymo, padėties medžio struktūroje, identifikatoriaus, konteksto ir reikšmių medžio žymes, minėti identifikatorius, kontekstas, ir kiekviena atskira reikšmė yra minėtos išraiškos elementai.

22. Redukuojantį procesorių pagal Apibrėžtis 8 - 21, apibūdinamą tuo, kad įjungtas į minėtą išraišką reikšmių medis įjungia galinius elementus ir sudėtinius elementus, kiekvienas sudėtinis elementas įjungia būseną ir reikšmių sąrašą.

23. Redukuojantį procesorių pagal Apibrėžtis 8 - 22, apibūdinamą tuo, kad minėtos išraiškos minėtas vykdymo pažymėjimas turi mažiausiai dvi būsenas, iš kurių pirmoji yra neveiklumo būsena ir antroji - vykdymo būsena.

24. Redukuojantį procesorių pagal Apibrėžtis 8 - 23, apibūdinamą tuo, kad minėtos išraiškos minėtas padėties pažymėjimas turi mažiausiai dvi būsenas, iš kurių pirmoji yra mazgo pozicija ir antroji - šaknies pozicija.

25. Redukuojantį procesorių pagal Apibrėžtis 8 - 24, apibūdinamą tuo, kad minėtos išraiškos kiekvienas elementas yra sudarytas iš požymių žodžio ir skaitmeninio žodžio, minėtas skaitmeninis žodis sudarytas iš tam tikro bitų, esančių 'teisinga' arba 'neteisinga' būsenose, skaičiaus.

26. Redukuojantį procesorių pagal Apibrėžtis 12 - 25, apibūdinamą tuo, kad minėti požymių žodžiai priklauso netiesioginių ir tiesioginių žodžių klasėms.

27. Redukuojantį procesorių pagal Apibrėžtis 12 - 26, apibūdinamą tuo, kad jis yra pritaikytas naudoti pirmąjį kodavimo tipą dvejetainiam žodžiui, atspindinčiam sveiką skaičių ir antrąjį kodavimo tipą dvejetainiam žodžiui, atspindinčiam plaukiojančio kablelio skaičių, ir kad minėti kodavimai yra

tokie, kad plaukiojančio kablelio skaičiaus atspindėjimas atitinka sveiko skaičiaus atspindėjimą.

28. Redukuojantį procesorių pagal Apibrėžtis 8 - 27, apibūdinamą tuo, kad sveikas skaičius yra pritaikytas saugoti minėtame elemente dvejetainiame formate taip, kad visos reikšmės eilėje nuo mažiausio iki didžiausio skaičiaus susideda iš kelių bitų reikšmių, ir kur nulis yra atitinka šios eilės vidurį, o jo dvejetainis formatas turi 'teisinga' reikšmingiausiame bite ir 'neteisinga' likusiuose bituose.

29. Redukuojantį procesorių pagal Apibrėžtis 8 - 28, apibūdinamą tuo, kad dvejetainė plaukiojančio kablelio reikšmė įjungia ženklą, eksponentės ženklą ir kodo lauką, eksponentės lauką ir mantisės lauką, minėti ženklas, eksponentės ženklas ir kodo laukas indikuoja į dalinimo ribą tarp minėto eksponentės lauko ir minėto mantisės lauko, ir todėl eksponentė ie mantisė turi varijuojančius ilgius.

30. Redukuojantį procesorių pagal Apibrėžtį 29, apibūdinamą tuo, kad minėti žodžiai, atspindintys skaitmeninių reikšmių žodžius, yra pateikti dažnuminiame kodavime, t. y. kiekvienas koduotas reikšmės atspindėjimas atitinka tik vienai interpretuojamai reikšmei.

31. Redukuojantį procesorių pagal Apibrėžtis 8 - 30, apibūdinamą tuo, kad mažiausiai keli minėtos išraiškos pažymėjimai yra skirtingose būsenose, kiekvieną būseną atitinka kelių bitų kodas.

32. Redukuojantį procesorių pagal bet kurias prieš tai sekusias Apibrėžtis, apibūdinamą turintį skaitmeninį aritmetinį įrenginį, skirtą vykdyti aritmetines, logines ir santykių operacijas su skaitmeninių reikšmių elementais, ir įjungiantį:

a) įėjimą, įjungiantį magistralių rinkinį, kiekvienos magistralės veikla surišta su sąrašo elementu, esančiu sąraše, minėtas sąrašas įjungia instrukcijos informaciją apie žodžius, esančius sąraše,

b) skaičiavimo priemones, prie kurių prijungtos minėtos magistralės, ir kurios vykdo operacijas su minėtais žodžiais minėtame sąraše panaudodamos minėtą instrukcijos informaciją

perrašant minėtus žodžius pagal minėtas instrukcijas, ir

c) išėjimą, pritaikytą perrašyto rezultato išvedimui, kartu įjungiantį tokio pačio dydžio ir konfigūracijos, kaip ir įėjime, magistralių rinkinį.

33. Redukuojantį procesorių pagal Apibrėžtį 32, a p i b ū d i n a m ą tuo, kad minėto sąrašo mažiausiai vienas elementas yra rezervuotas instrukcijos informacijai ir yra perduodamas tam tikromis minėto įėjimo magistralėmis, o skaitmeninė reikšmė, kuri turi būti apdorota, yra perduodama kitomis minėto įėjimo magistralėmis, ir minėtos skaičiavimo priemonės yra pritaikytos vykdyti operacijas, perrašant minėto įėjimo sąrašo skaitmenines reikšmes.

34. Redukuojantį procesorių pagal Apibrėžtį 32, a p i b ū d i n a m ą tuo, kad keletas grandinių, kurių kiekviena yra pritaikyta atlikti tam tikras operacijas su įėjimo magistralėse esančiomis skaitmeninėmis reikšmėmis, gali pateikti savo rezultatus lygiagrečiai, ir kad valdymo priemonės (1a; 27), aprūpintos minėta instrukcijos informacija, tarp visų gautų rezultatų išsirenka rezultatą(us), atitinkančius vykdomą istrukciją.

35. Redukuojantį procesorių pagal Apibrėžtį 34, a p i b ū d i n a m ą tuo, kad kai minėtas sąrašas yra pritaikytas įjungti funkciją, kurios vienas iš elementų yra instrukcijos kodas ir likę yra minėtos instrukcijos argumentai, minėtos skaičiavimo priemonės vykdo instrukciją perrašant ir ciklišKai pernešant minėtą instrukcijos kodą iš minėto išėjimo į minėtą įėjimą, kol gaunamas galutinis rezulatatas, ir po kiekvieno perrašymo minėtos skaičiavimo priemonės pritaikomos perrašyti minėtą sąrašą įjungiant modifikuotą instrukcijos kodo žodį, jeigu tai reikalinga vykdomiems skaičiavimams, po kurio, jeigu reikalinga, seka reikšmių žodžiai.

36. Redukuojantį procesorių pagal bet kurias prieš tai sekusias Apibrėžtis, a p i b ū d i n a m ą tuo, kad tam tikras pirmos eilės procesorių kiekis yra sujungti tarpusavyje tinklo pagalba, minėti sujungti pirmos eilės procesoriai sudaro antros eilės redukuojantį procesorių.

37. Redukuojantį procesorių pagal bet kurias prieš tai sekusias

Apibrėžtis, a p i b ū d i n a m ą tuo, kad kiekvieno pirmos eilės redukuojančio procesoriaus viena ar kelios minėtos atminties ląstelės yra pritaikytos saugoti išraišką, kuri yra vykdymo, pozijos medžio struktūroje, identifikatoriaus, konteksto ir reikšmių medžio žymėjimas, minėti identifikatorius, kontekstas, ir kiekviena atskira reikšmė yra minėtos išraiškos elementai, kiekvienas elementas minėtoje išraiškoje sudarytas iš požymių ir skaitmeninio žodžių, minėtas skaitmeninis žodis yra sudarytas iš tam tikro bitų, kurie gali būti 'teisinga' ir 'neteisinga', skaičiaus ir minėti požymių žodžiai yra skirstomi į netiesioginius ir tiesioginius, ir minėto netiesioginio tipo elemento bitinis piešinys yra sudalintas į srities ir adreso laukus.

38. Redukuojantį procesorių pagal Apibrėžtį 37, a p i b ū d i n a m ą tuo, kad keletas pirmos eilės redukuojančių procesorių ($FOP_{1,1}$' iki $FOP_{M,M}$') yra sujungti į kvadratinę matricą sudarydami antros eilės redukuojantį procesorių, ir kiekvienas pirmos eilės procesorius minėtoje kvadratinėje matricoje turi kanalą (CAN) nueinantį į kiekvieną kaimyninį pirmos eilės procesorių minėtoje kvadratinėje matricoje (pieš. 37A, 37C ir 37D).

39. Redukuojantį procesorių pagal Apibrėžtį 38, a p i b ū d i n a m ą tuo, kad minėtas antros eilės redukuojantis procesorius yra sudalintas į logines sritis (pieš. 37C), kurių apimtis yra 2^n dauginta iš 2^n pirmos eilės redukuojančių procesorių dydžio, minėtos loginės sritys patalpintos viena šalia kitos taisyklinga tvarka, perdengdamos visą minėto antros eilės redukuojančio procesoriaus minėtą kvadratinę matricą.

40. Redukuojantį procesorių pagal Apibrėžtį 39, a p i b ū d i n a m ą tuo, kad minėti pirmos eilės procesoriai yra sujungti tarpusavyje į minėtą antros eilės procesorių taip, kad kiekviena minėtos antros eilės procesoriaus minėta sritis yra perkeliama vidinio pertvarkymo mechanizmo pagalba per pusę srities dydžio bet kuria kryptimi minėtame antros eilės procesoriuje.

41. Redukuojantį procesorių pagal Apibrėžtis 39 arba 40, a p i b ū d i n a m ą tuo, kad mažiausiai vienas adresuojamas netiesioginis elementas, įjungiantis adresą, yra saugomas kiekvienoje srityje, minėtas elementas yra pasiekiamas tik atminties ląstelėms, kurios priklauso duotai sričiai.

42. Redukuojantį procesorių pagal Apibrėžtis 37 - 41, apibūdinamą tuo, kad keletas minėtų pirmos eilės redukuojančių procesorių yra sujungti į tinklų hierarchinę sistemą (NET_1 iki NET_n), kur kiekvienas tinklas yra magistralė.

43. Redukuojantį procesorių pagal Apibrėžtis 37 - 42, apibūdinamą tuo, kad keletas minėtų pirmos eilės redukuojančių procesorių yra sujungti į tinklų hierarchinę sistemą, kur kiekvienas tinklas yra žiedinio pobūdžio.

44. Redukuojantį procesorių pagal Apibrėžtis 37 - 43, apibūdinamą tuo, kad keletas minėtų pirmos eilės redukuojančių procesorių yra sujungti į tinklų hierarchinę sistemą, mažiausiai du tinklų tipai yra naudojami, pirmasis tipas yra magistralė, antrasis tipas yra žiedinio pobūdžio ir trečiasis tipas yra kvadratinė matrica.

45. Redukuojantį procesorių pagal bet kurias prieš tai sekusias Apibrėžtis, apibūdinamą tuo, kad naudojamos mažiausiai vieno porto įranga, prijungta prie minėtos aktyvios atminties priemonių, ir mažiausiai vieno konteksto priemonės prijungtos prie minėto mažiausiai vieno porto įrangos.

46. Redukuojantį procesorių pagal Apibrėžtį 45, apibūdinamą tuo, kad naudojamos priemonės, skirtos signalų sekos, ateinančios iš porto įrangos, sulyginimui su mažiausiai vienoje atminties ląstelėje saugomoma seka, minėta saugoma seka turi galimą neapibrėžtą sekos elementą ($) minėtos aktyvios atminties priemonėse, ir priemones (CU, 3), skirtas saugomos sekos perrašymui į niekur, t. y. į kažką atspindintį neatitikimą, jeigu sulyginant gaunamas aiškus skirtumas, arba minėtos saugomos sekos perrašymui į tam tikrą seką, kuri yra signalo ir saugomos sekų unifikacija.

47. Redukuojantį procesorių pagal Apibrėžtį 46, apibūdinamą tuo, kad minėtos sulyginimo priemonės vykdo nustatyto skaičiaus sąrašo elementų grupių sulyginimą.

48. Redukuojantį procesorių pagal Apibrėžtis 46 arba 47, apibūdinamą tuo, kad naudojamos priemonės (3, during; 7, 8, 9) pateikiant minėtą signalo seką kaip diskretizuotą signalą (pieš. 34), kuris keičiasi laiko bėgyje per atskirus kvantavimo periodus, minėta signalo seka yra elementų grupių sąrašas, kur

kiekviena grupė įjungia laiko trukmę ir mažiausiai vieną signalo amplitudę šios trukmės metu.

49. Redukuojantį procesorių pagal Apibrėžtį 48, apibūdinamą tuo, kad minėtas nustatytas sąrašo elementų skaičius kiekvienoje grupėje yra du, pateikiami poromis, ir kiekviena pora įjungia laiko ir signalo amplitudės kombinaciją.

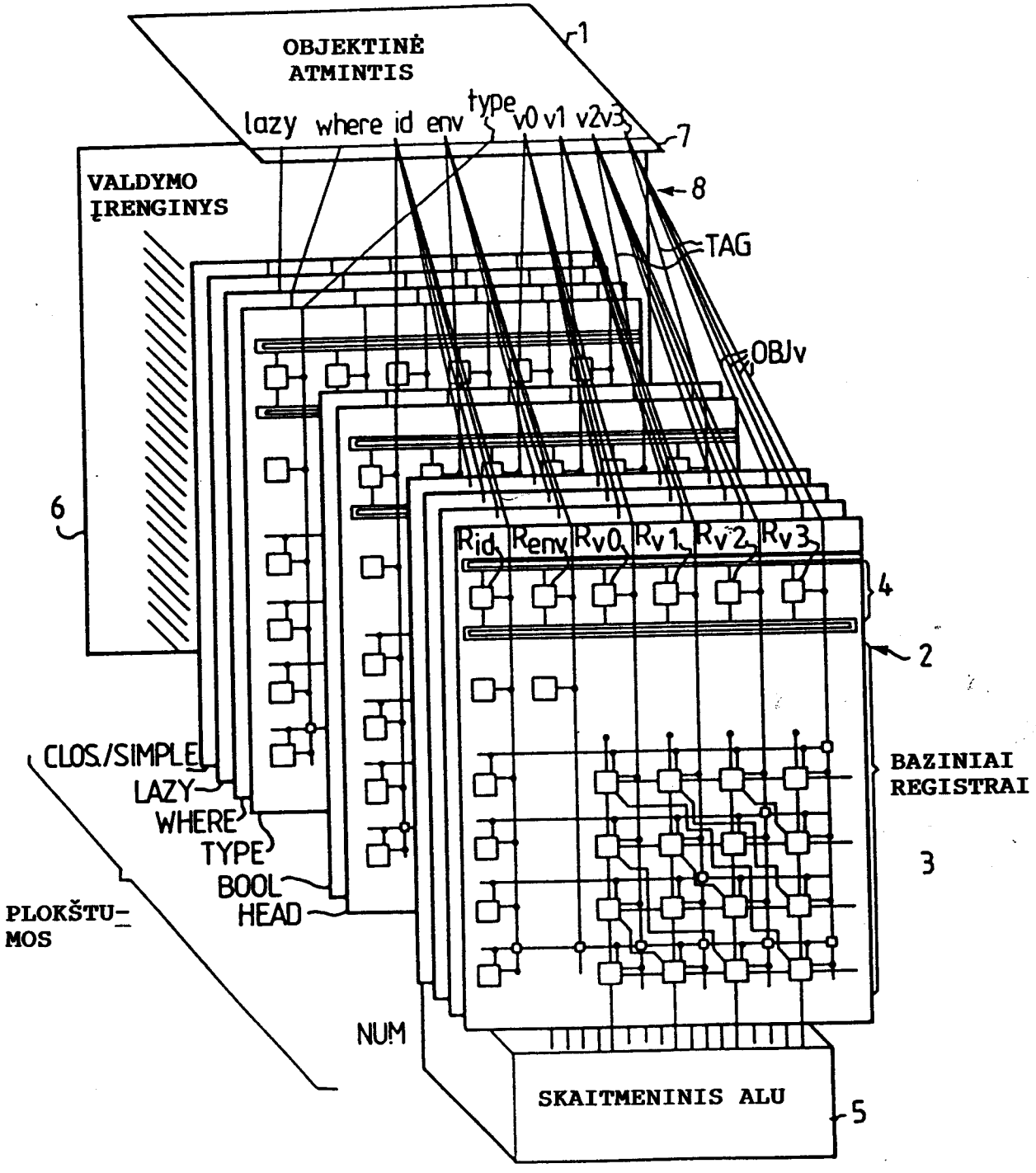
50. Redukuojantį procesorių pagal Apibrėžtis 45 - 49, apibūdinamą tuo, kad minėtos objektinės atminties ląstelės yra sujungtos į struktūrą minėtoje aktyvioje atmintyje pritaikytą kompjuterio programai, realizuotai abstrakčios sintaksės su eksplikuojančiu ar implikuojančiu kodu forma, minėta sintaksė aprašo skirtingus abstrakčius objektus išraiškų pagalba, kiekvienos objektinės atminties ląstelės priemonės gali vienu metu saugoti mažiausiai vienos minėtos sintaksinės išraiškos dalį atitinkamų duomenų ar/ir programos struktūros formoje.

51. Redukuojantį procesorių pagal Apibrėžtį 44, apibūdinamą tuo, kad daugybė geografiškai išskaidytų antros eilės redukuojančių procesorių (GSOP) sujungti vienas su kitu, sudaro trečios eilės redukuojantį procesorių.

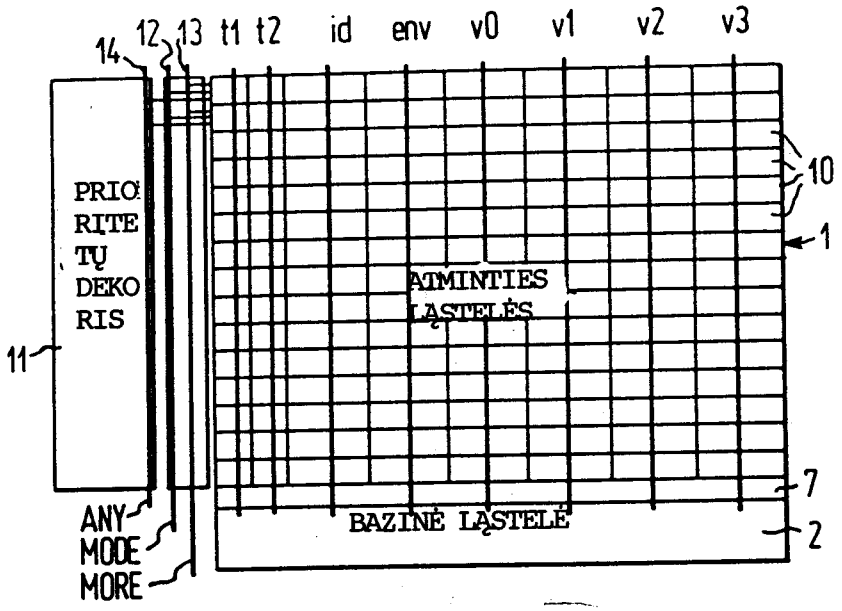
52. Redukuojantį procesorių pagal Apibrėžtį 51, apibūdinamą tuo, kad antros eilės redukuojantis procesorius (SOP) yra pritaikytas sričių apribotam asociatyviniam adresavimui, ir kad trečios eilės redukuojantis procesorius yra pritaikytas fiziniam adresavimui.

53. Redukuojantį procesorių pagal Apibrėžtį 52, apibūdinamą tuo, kad kiekvienas geografiškai išskaidytas antros eilės redukuojantis procesorius (GSOP) esantis trečios eilės redukuojančiame procesoriuje, turi priemones tėvų, saugomų kituose GSOP, negu sūnus, atsekimui.

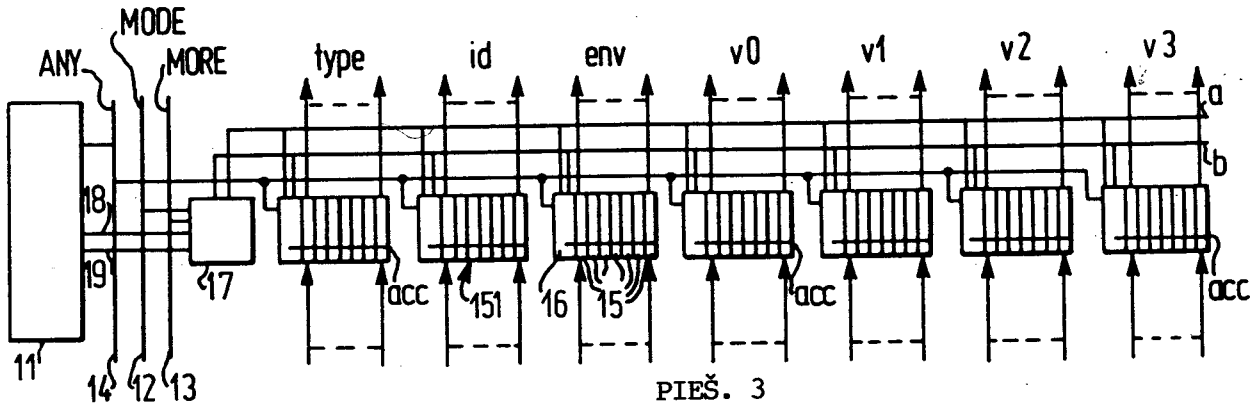
54. Redukuojantį procesorių pagal Apibrėžtį 53, apibūdinamą tuo, kad minėtos priemonės nelokalinių tėvų atsekimui turi lokalinėje objektinės atmintyje esančias duomenų struktūras, kurios kiekvienam sūnui, turinčiam nelokalinius tėvus, turi sąrašą su nelokalinių tėvų adresais, kuriuose yra dalis, identifikuojanti antros eilės procesorių (GSOP), saugontį nelokalinius tėvus.


OBJEKTINĖ ATMINTIS
1
lazy where id env type v0 v1 v2 v3
7
VALDYMO ĮRENGINYS
8
TAG
OBJv
6
Rid Renv Rv0 Rv1 Rv2 Rv3
4
2
CLOS/SIMPLE
LAZY
WHERE
TYPE
BOOL
HEAD
BAZINIAI REGISTRAI
3
PLOKŠTU-MOS
NUM
SKAITMENINIS ALU
5

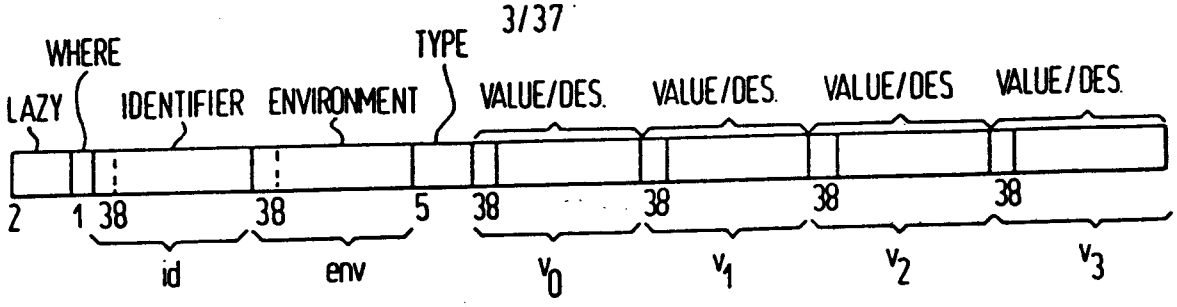
PIEŠ.1


14
12
13
t1
t2
id
env
v0
v1
v2
v3
PRIO
RITE
TŲ
DEKO
RIS
ATMINTIES
LĄSTELĖS
BAZINĖ LĄSTELĖ
10
1
7
2
11
ANY
MODE
MORE

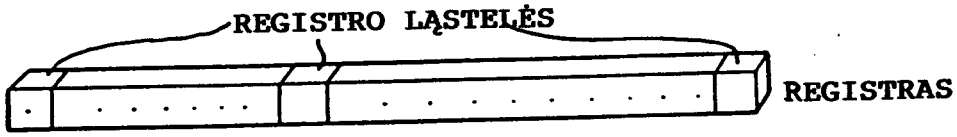
PIEŠ. 2


MODE
ANY
MORE
type
id
env
v0
v1
v2
v3
a
b
18
19
17
acc
151
16
15
acc
acc
11
14
12
13

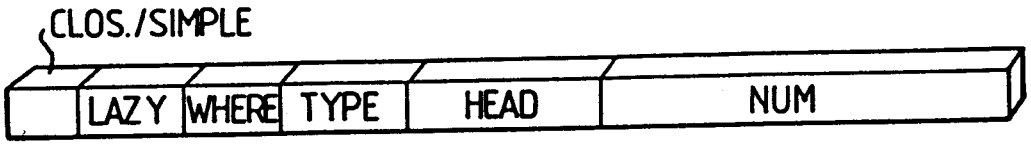
PIEŠ. 3



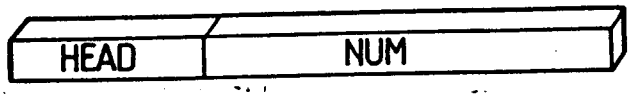
PIEŠ.4A



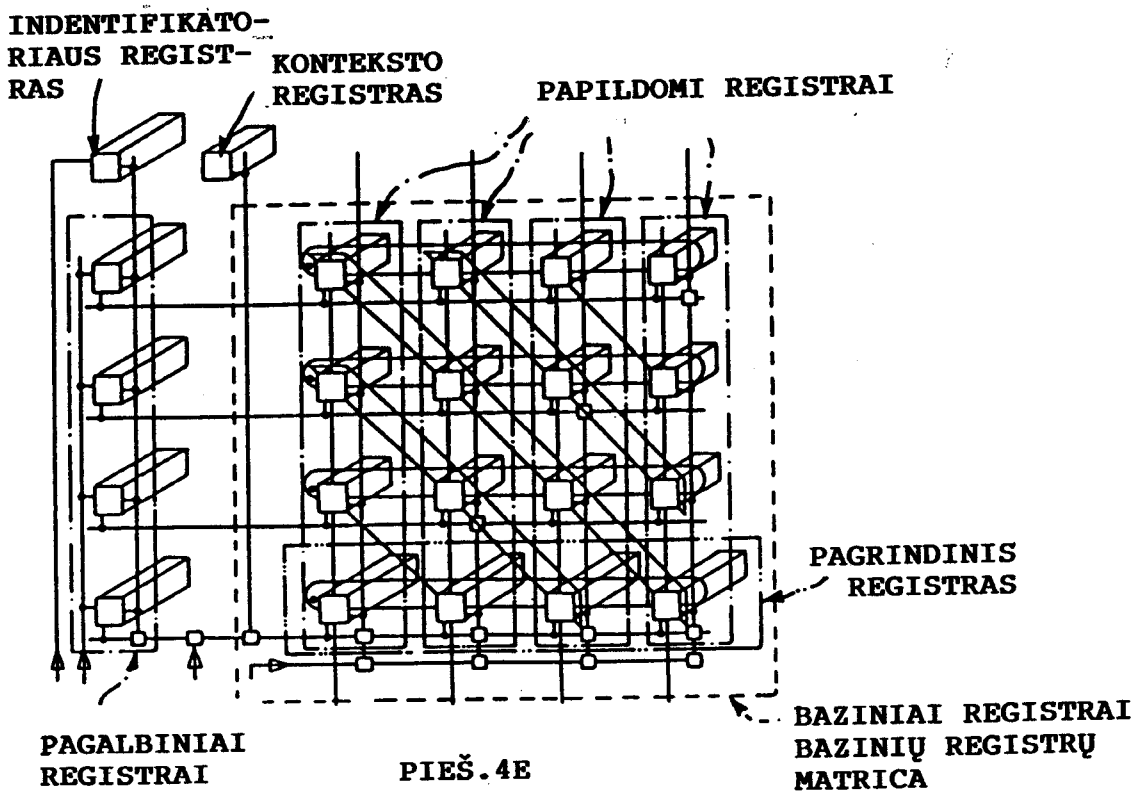
PIEŠ.4B



PILNAS REGISTRAS PIEŠ.4C

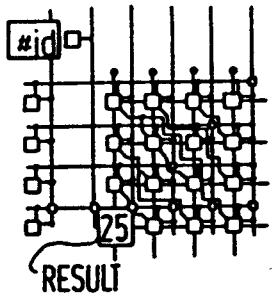


RIBOTAS REGISTRAS PIEŠ.4D



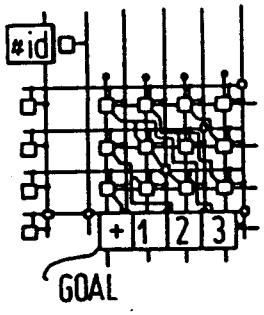
PIEŠ.4E

PAPRASTA REIKŠMĖ ARBA MAŠINOS IDENTI FIKATORIUS



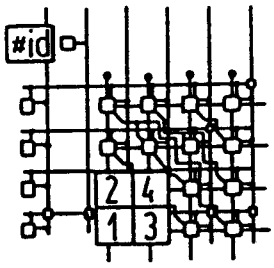
PIEŠ.5A

VIENO LYGIO STRUKTŪRA



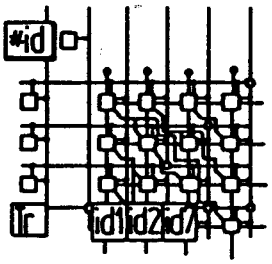
PIEŠ. 5B

DVIEJŲ LYGIŲ STRUKTŪRA



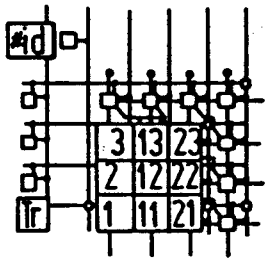
PIEŠ.5C

DVIEJŲ LYGIŲ STRUKTŪRA



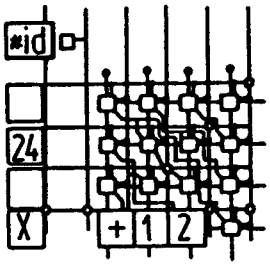
PIEŠ. 5D

TRIJŲ LYGIŲ STRUKTŪRA

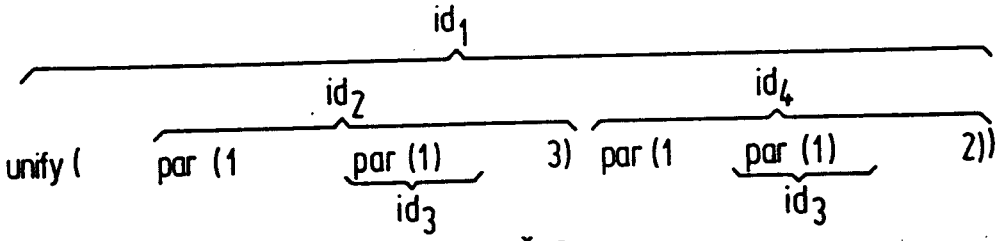


PIEŠ. 5E

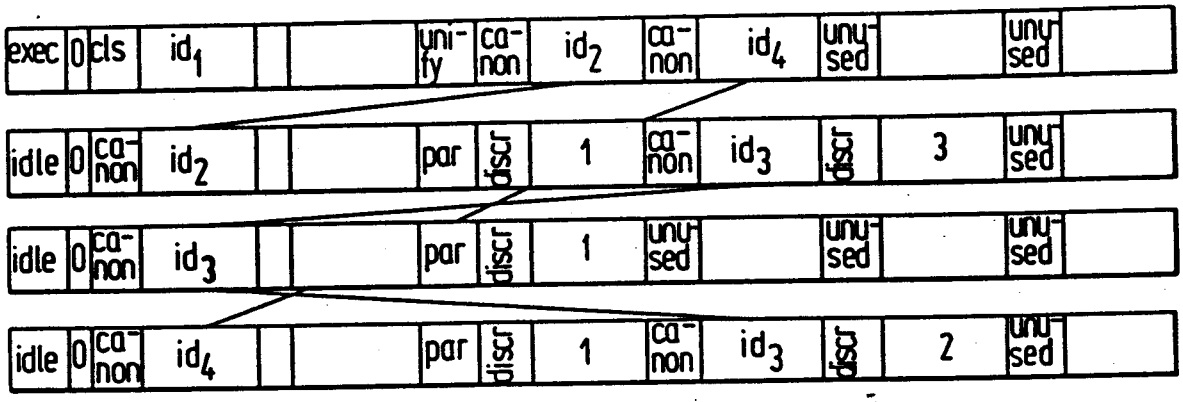
KONVEJERINIAME REŽIME



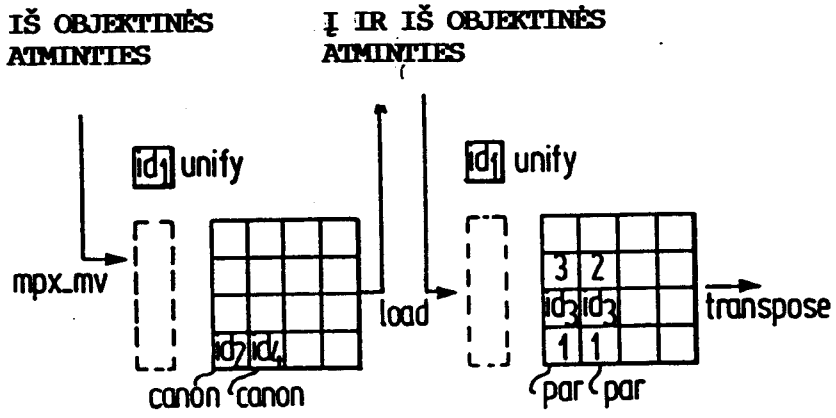
PIEŠ. 5F



PIEŠ.6A

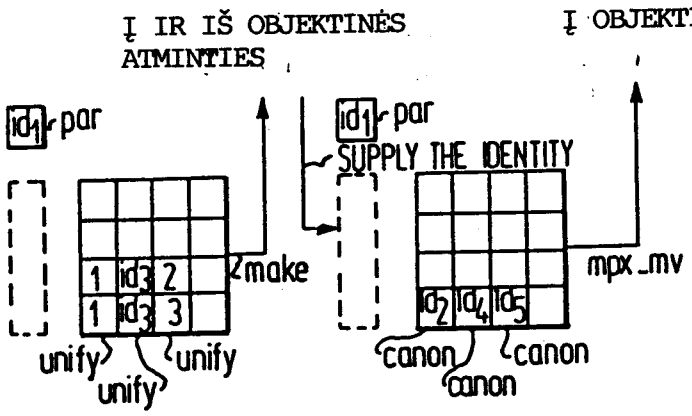


PIEŠ. 6B

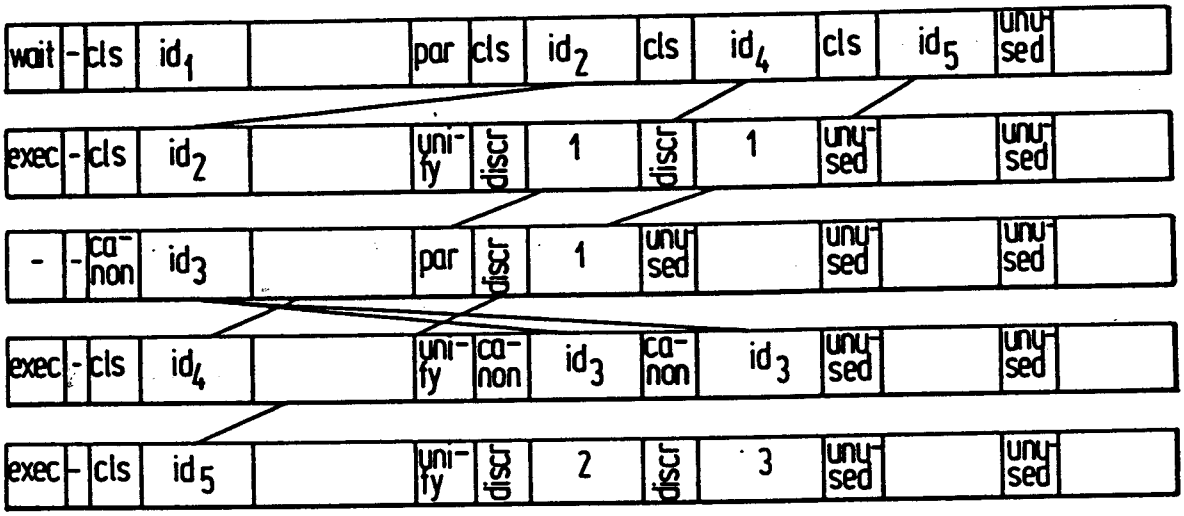


PIEŠ. 6C

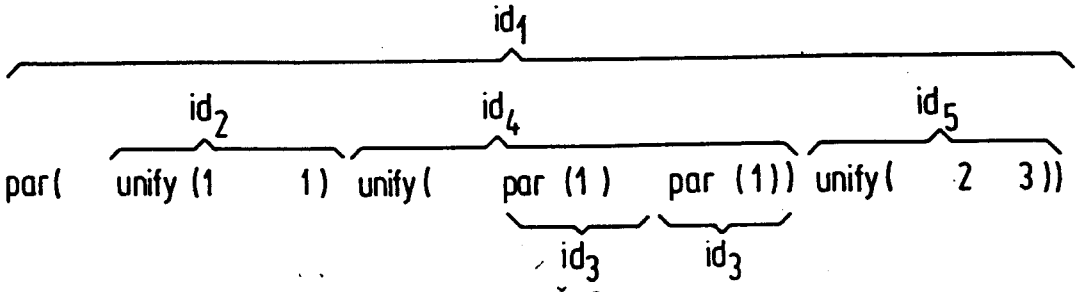
PIEŠ. 6D



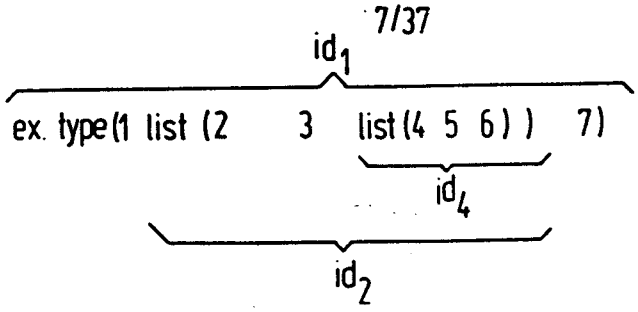
PIEŠ.6E PIEŠ. 6F



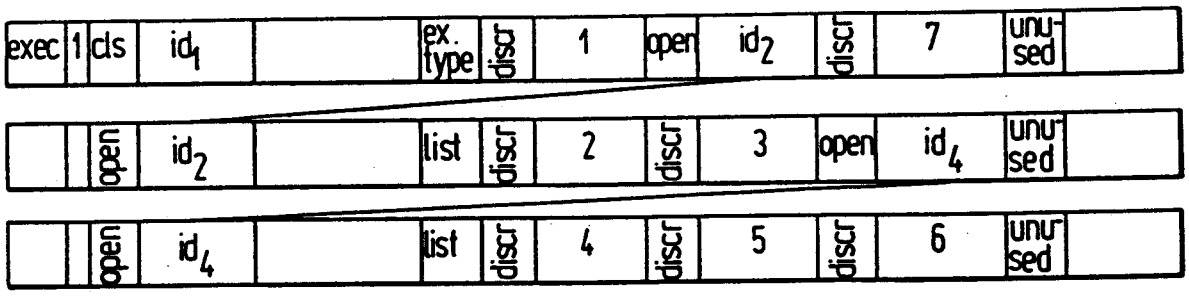
PIEŠ. 6G



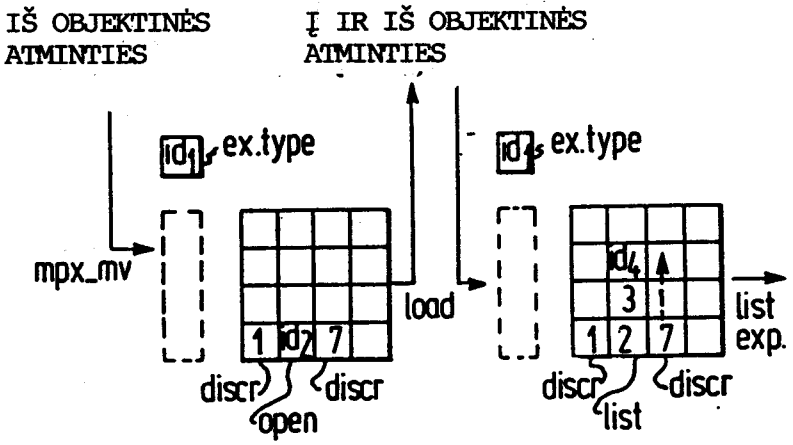
PIEŠ.6,H



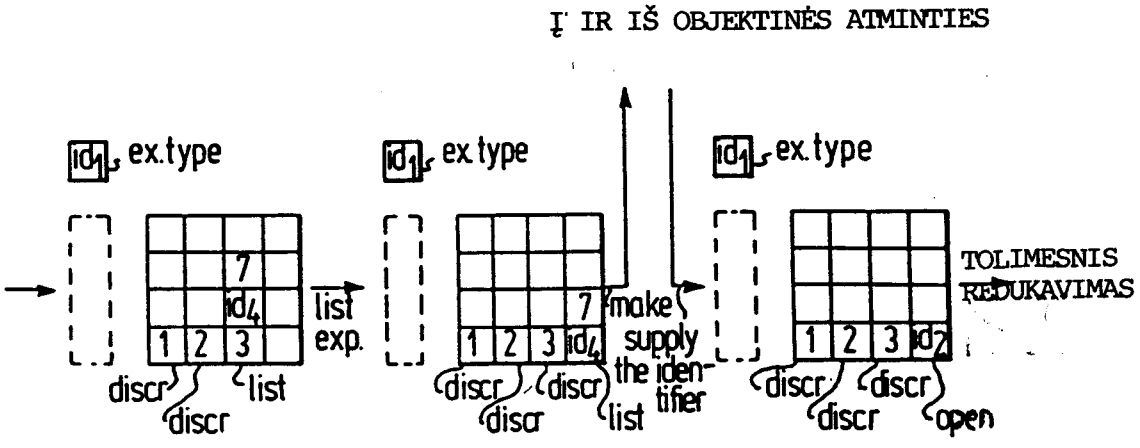
PIEŠ. 74



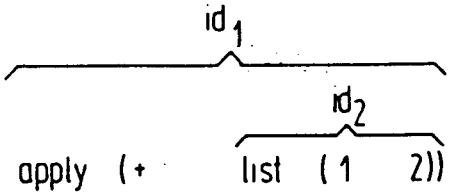
PIEŠ. 7B



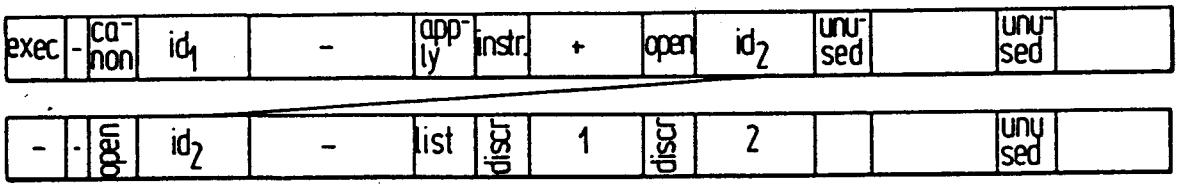
PIEŠ. 7C PIEŠ. 7D



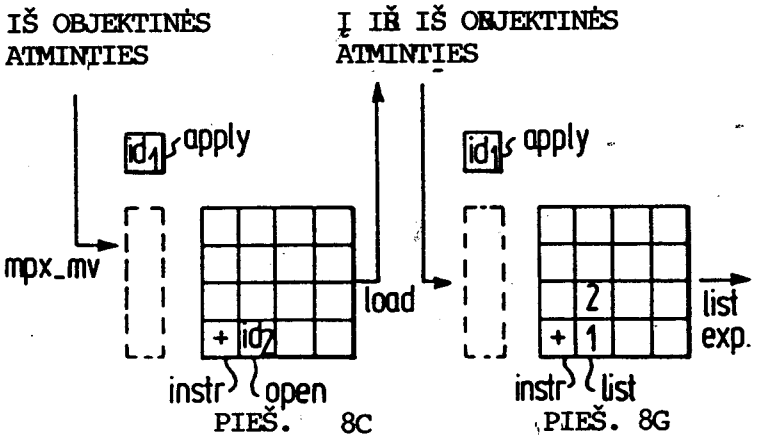
PIEŠ. 7E PIEŠ. 7F PIEŠ. 7G



PIEŠ. 8A

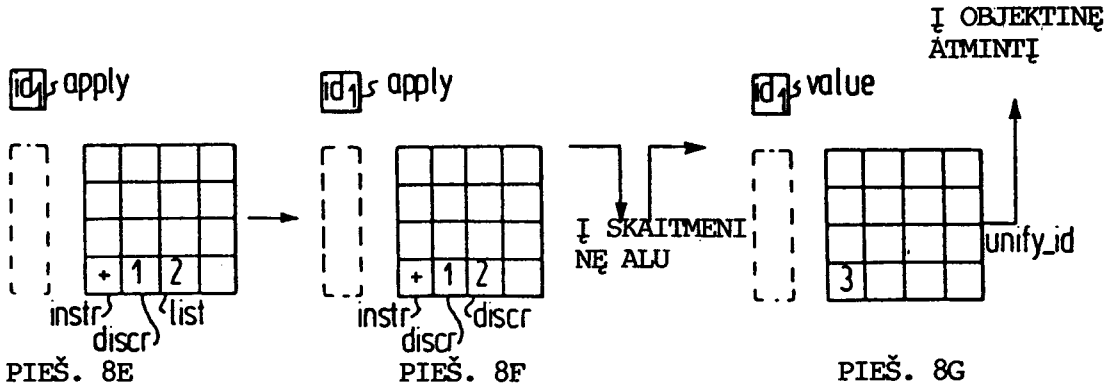


PIEŠ. 8B



PIEŠ. 8C

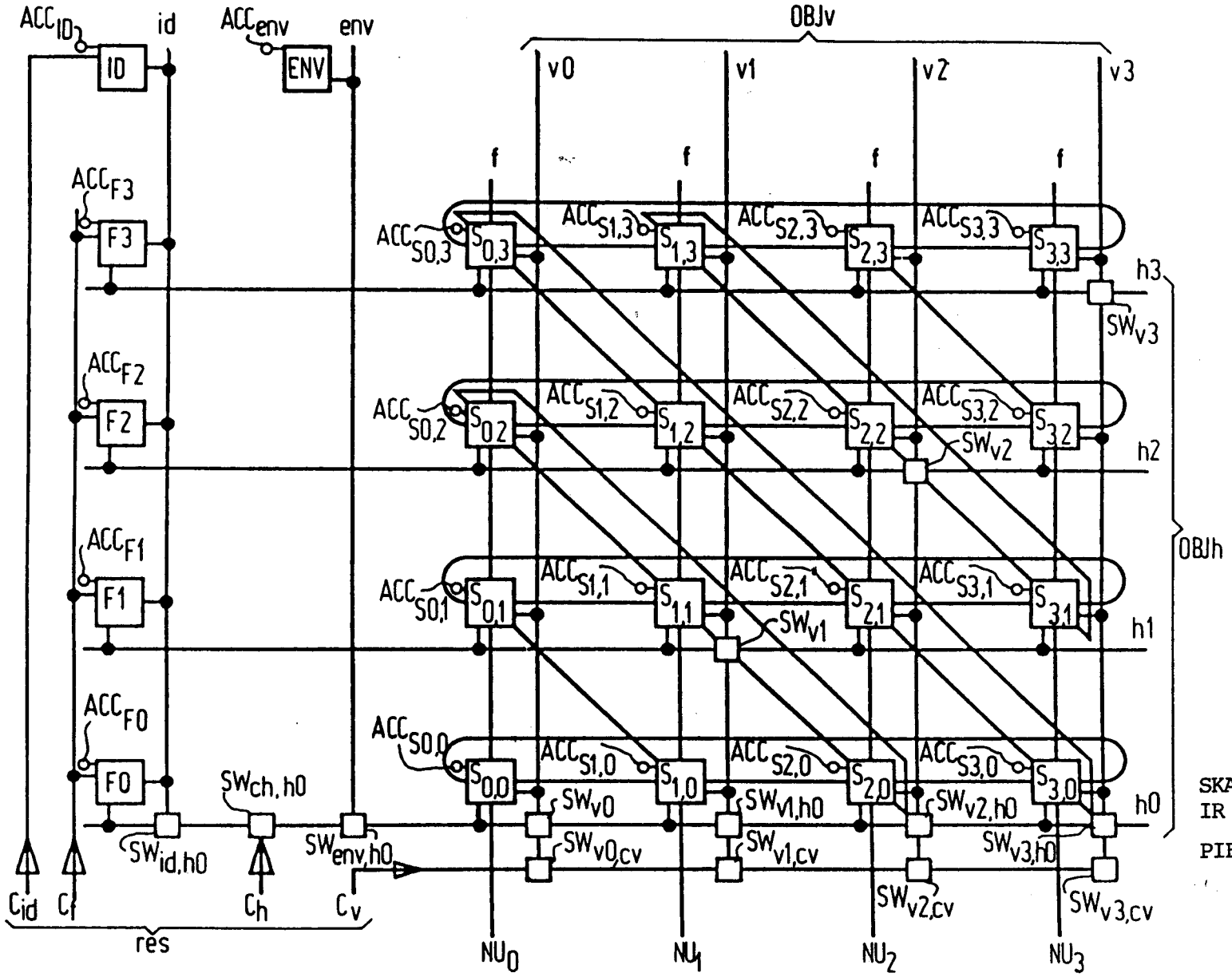
PIEŠ. 8G



PIEŠ. 8E

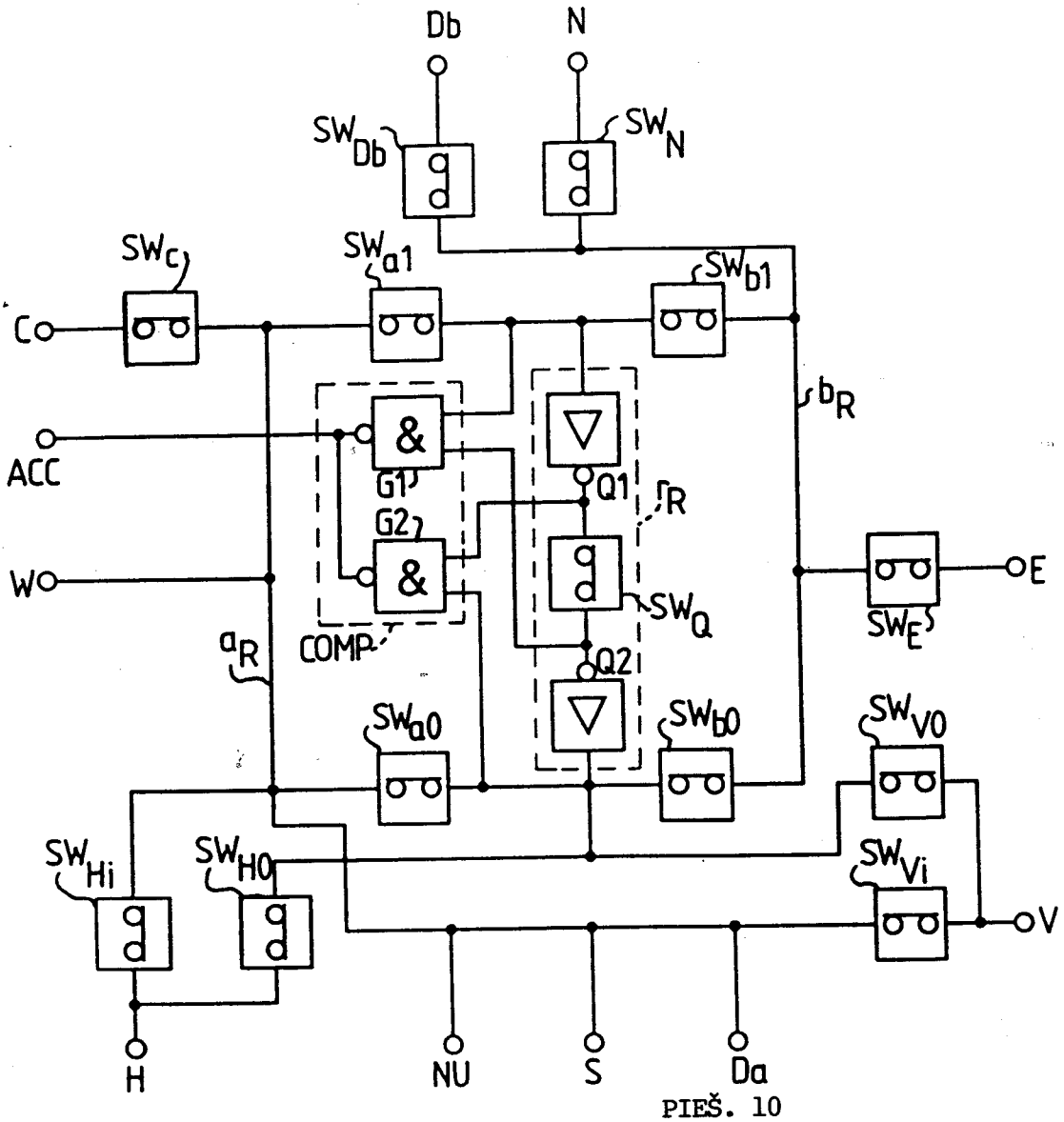
PIEŠ. 8F

PIEŠ. 8G



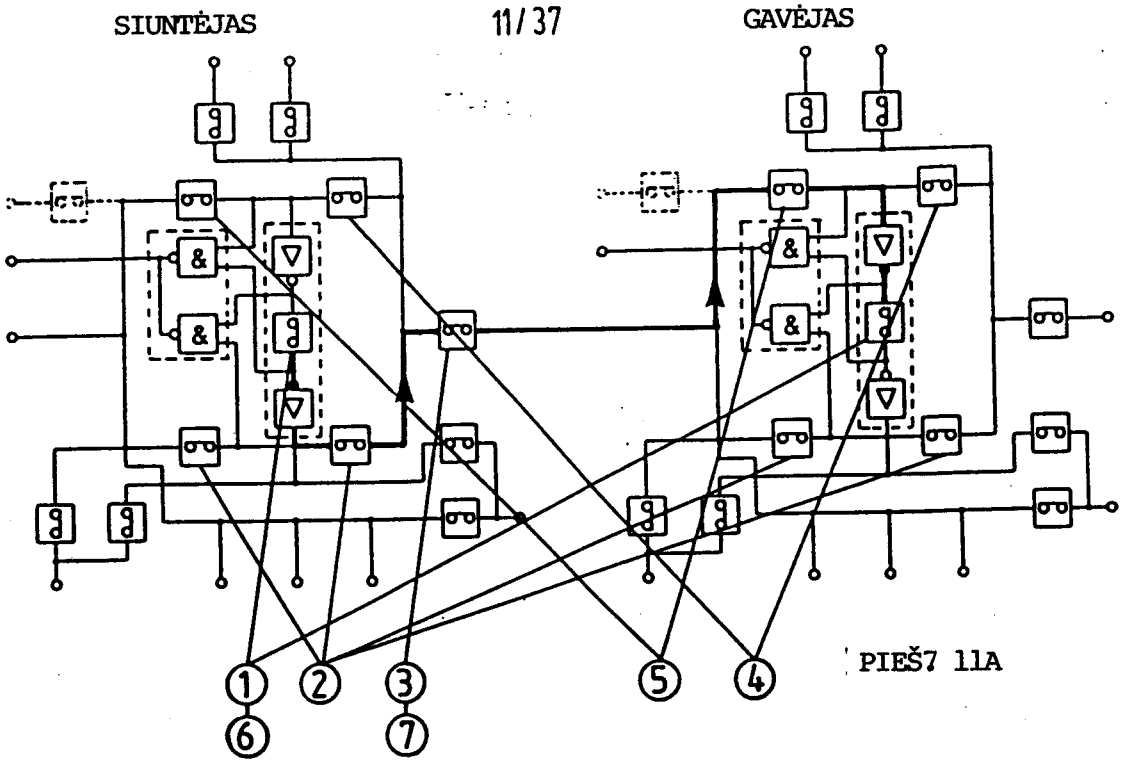
SKAITMENINĖ IR VIRŠŪNĖS PLOKŠTUMOS

PIEŠ. 9

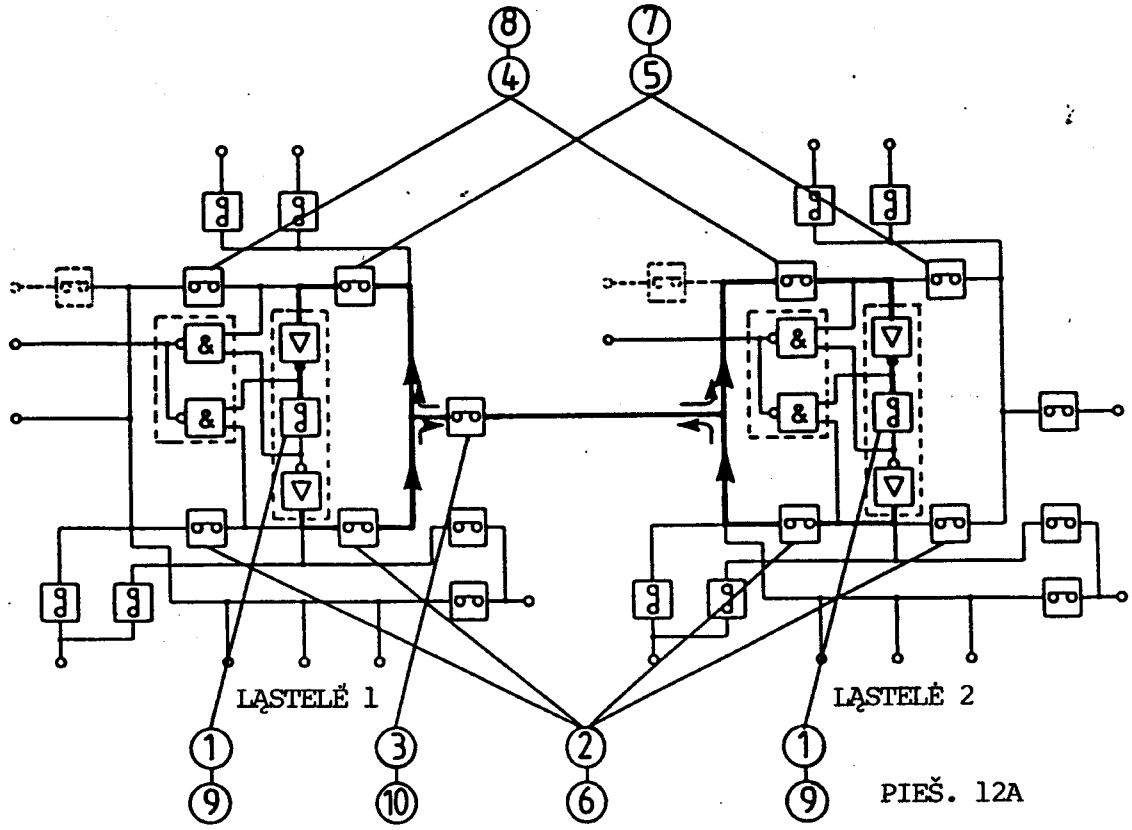


PIEŠ. 10

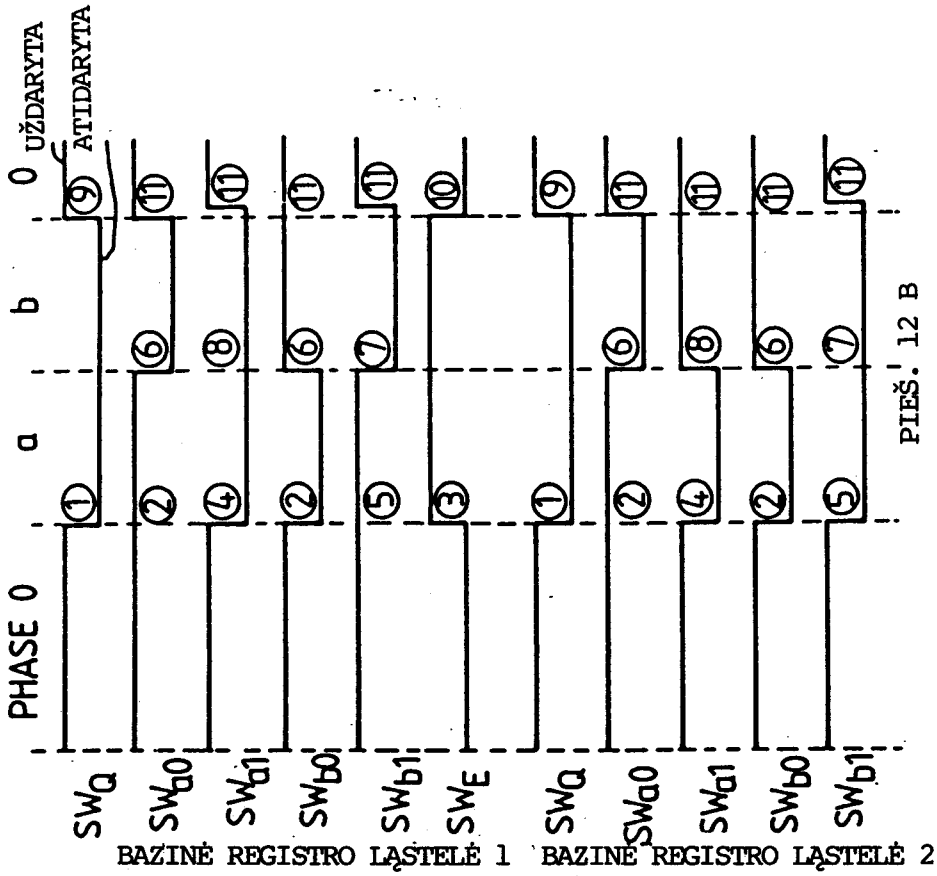
BENDRA REGISTRO LĄSTELĖ (SKAITMENINĖ IR VIRŠŪNĖS PLOKŠTUMOS)



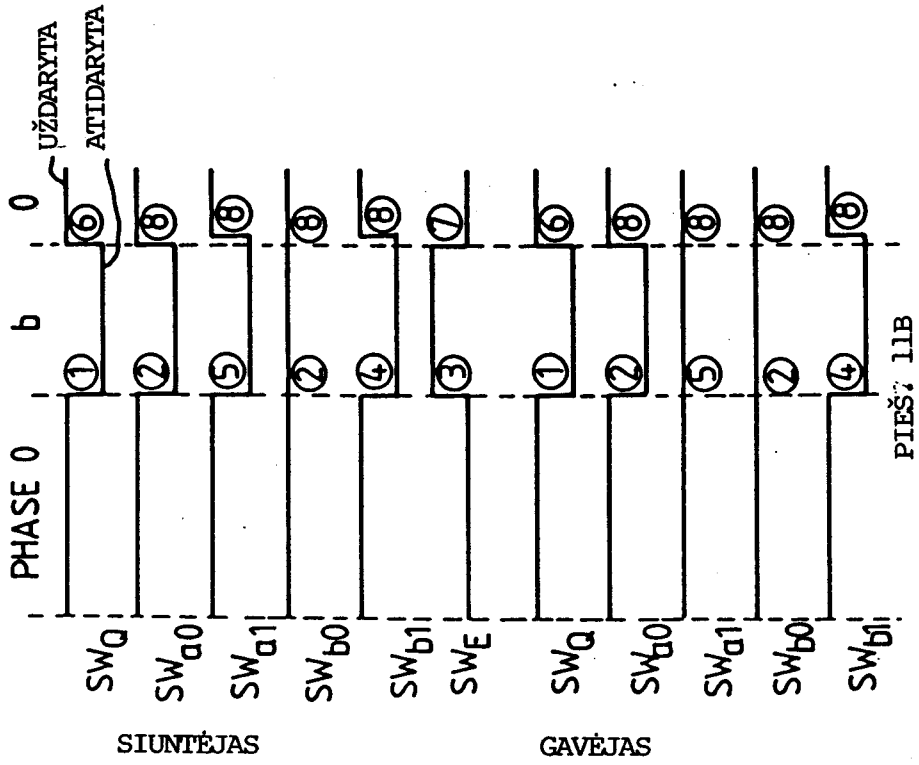
PIEŠ7 11A



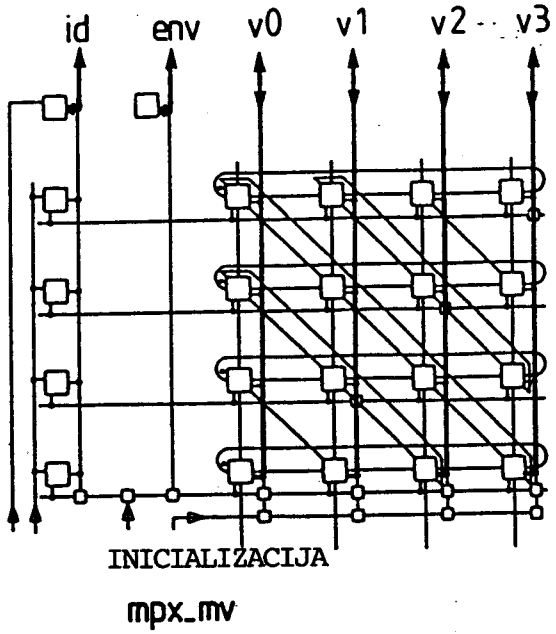
PIEŠ. 12A



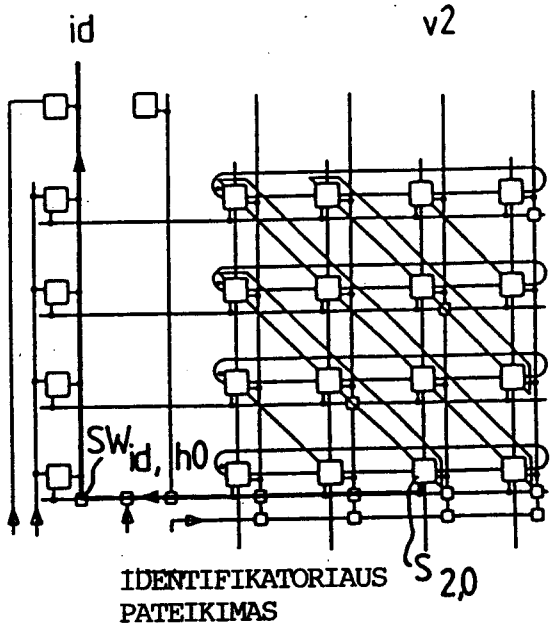
PIEŠ. 12 B



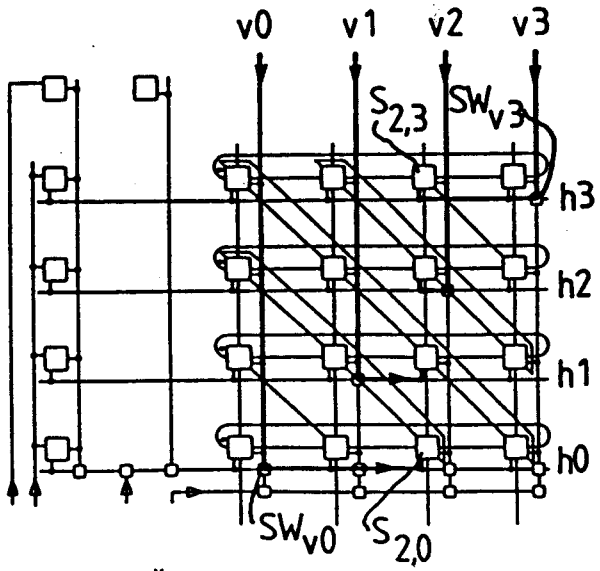
PIEŠ. 11B



PIEŠ. 13

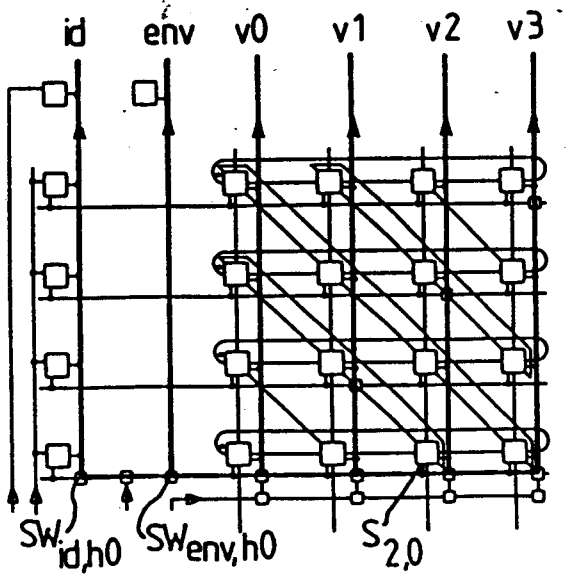


PIEŠ. 14



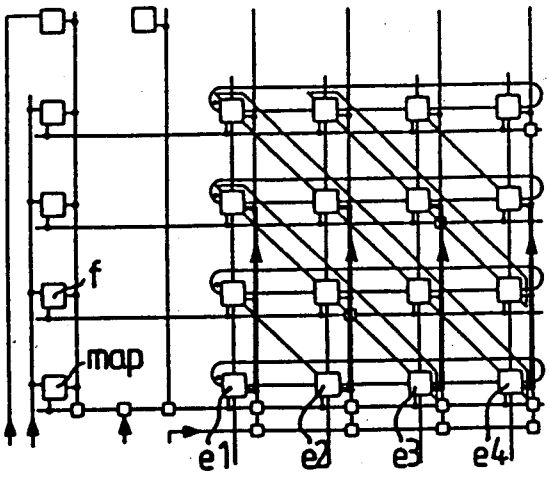
REIKŠMIŲ PAKROVIMAS Į A
REGISTRO STULPELĮ

PIEŠ. 15



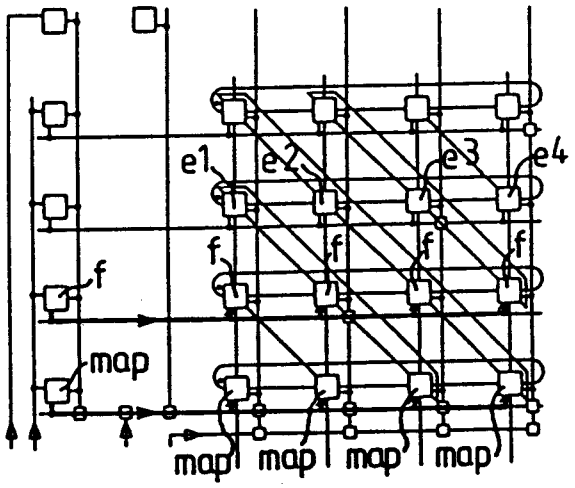
unify_id IDENTIFIKATORIAUS
PATEIKIMAS

PIEŠ. 16



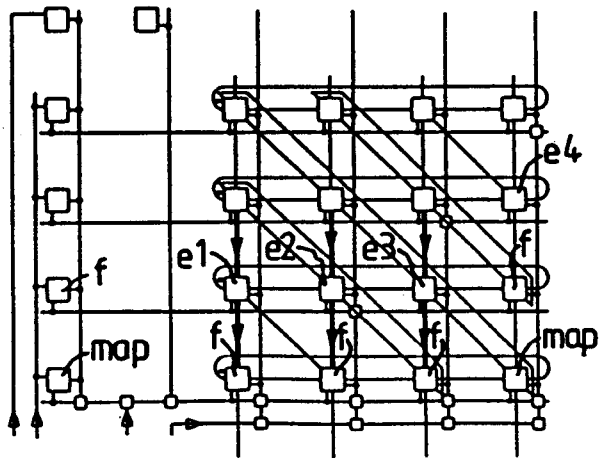
(map f (e1 e2 e3 e4))

PIEŠ. 17 A



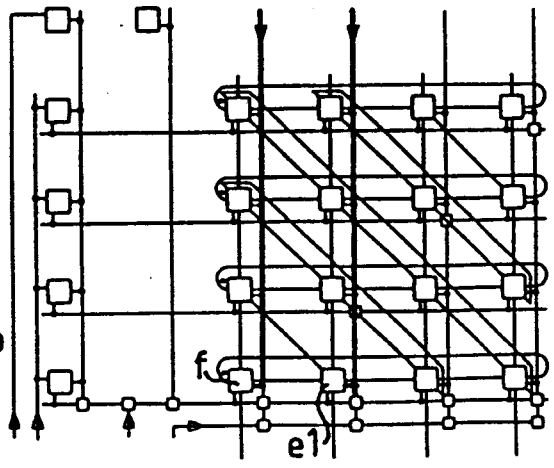
((map f e1)(map f e2)
(map f e3)(map f e4))

PIEŠ. 17 B



((f e1)(f e2)(f e3)
(map f e4))

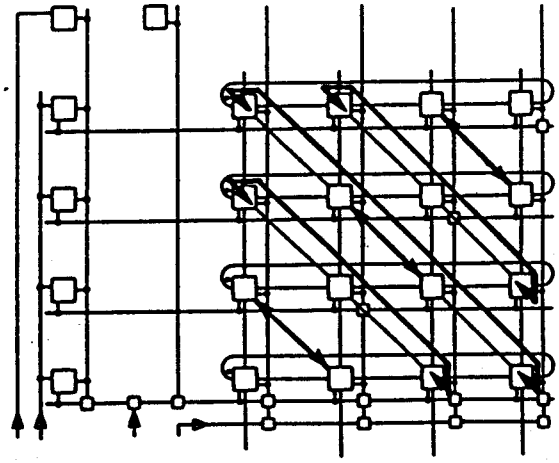
PIEŠ17 C



(f e1)

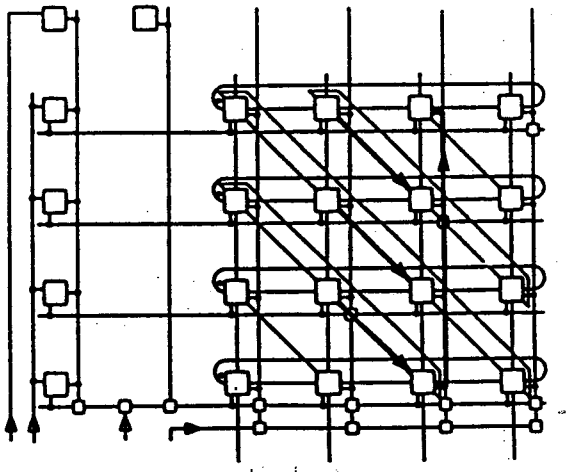
PERKRAUTA

PIEŠ. 17 D



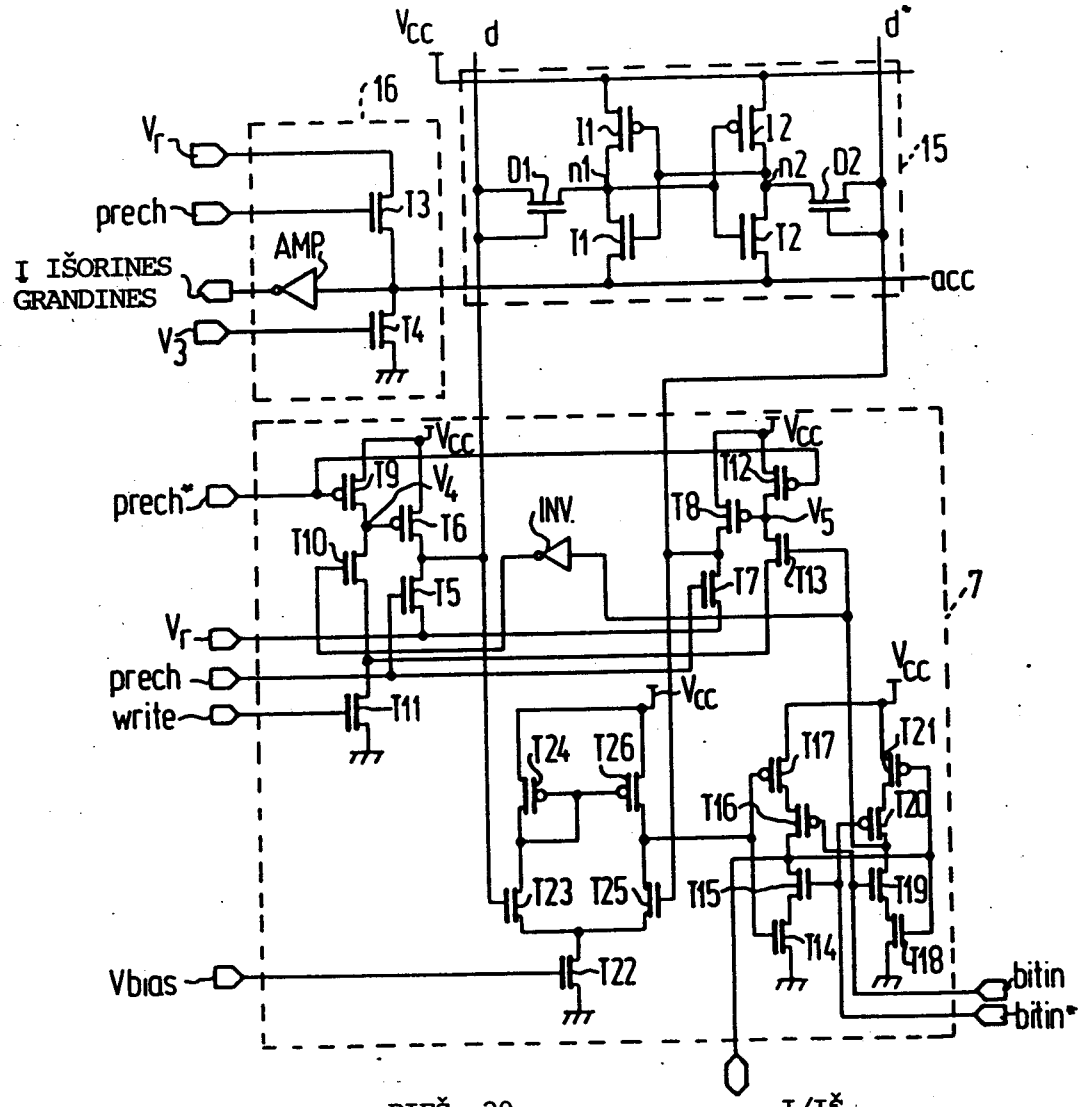
TRANSPONAVIMAS

PIEŠ. 18

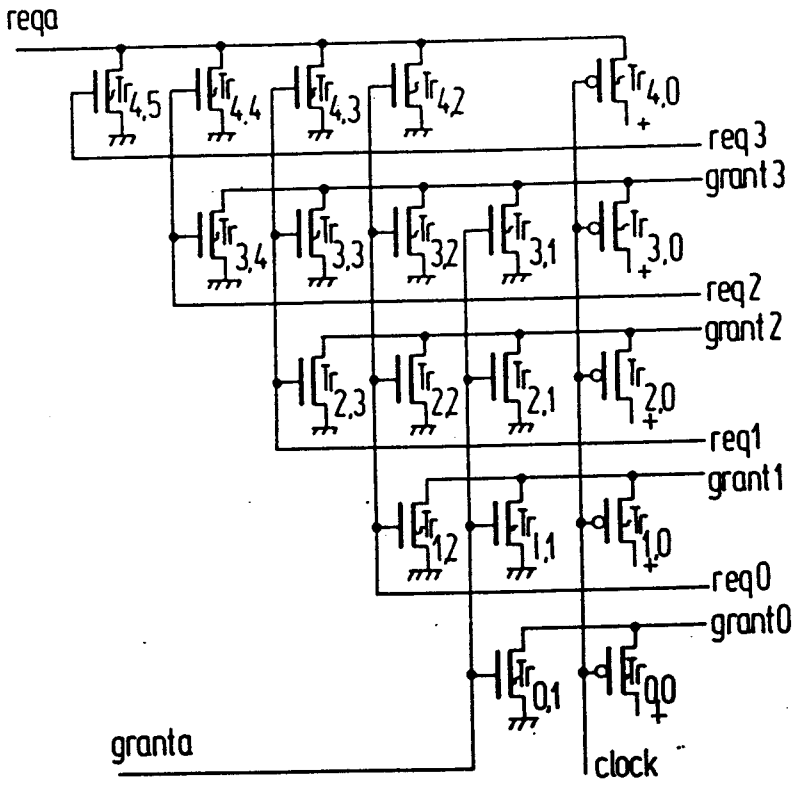


IŠPLEČIAMAS sąrašas

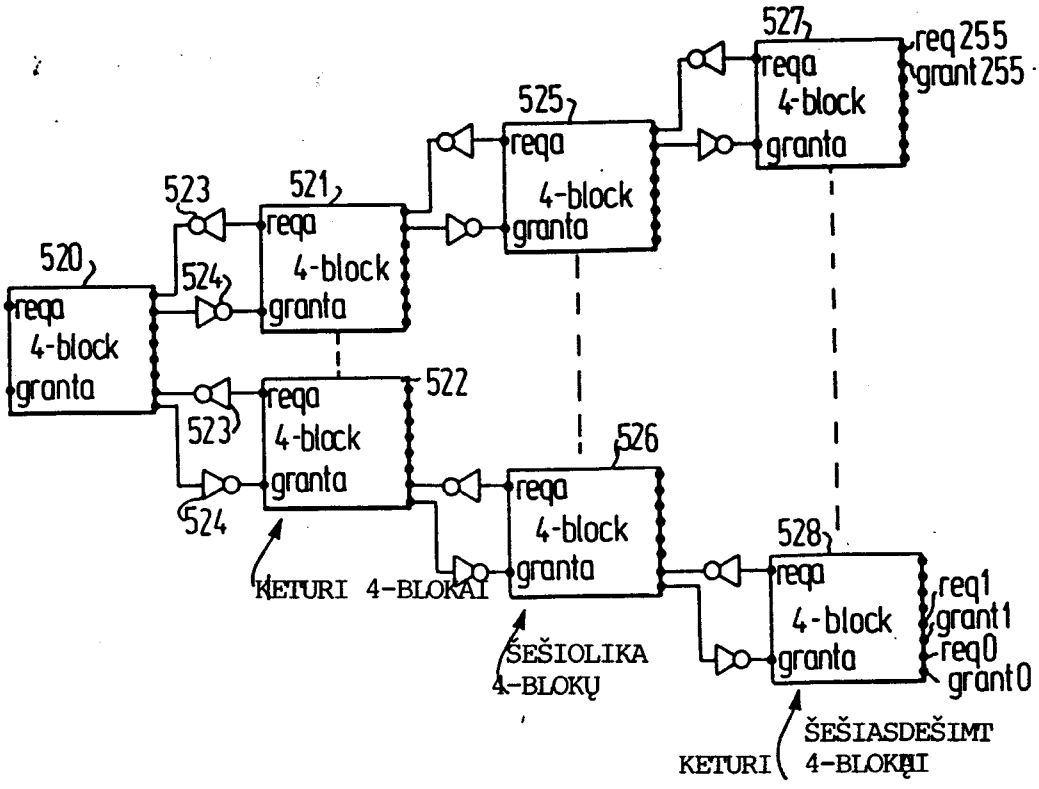
PIEŠ. 19


Vcc
d
d*
16
15
Vr
prech
I IŠORINES GRANDINES
AMP
V3
T3
T4
I1
I2
D1
n1
n2
D2
T1
T2
acc
prech*
T9
V4
T6
T10
T5
INV.
T8
T12
V5
T7
T13
7
write
T11
T24
T26
T17
T21
T16
T20
T23
T25
T15
T19
T14
T18
Vbias
T22
bitin
bitin*
Ą/IŠ

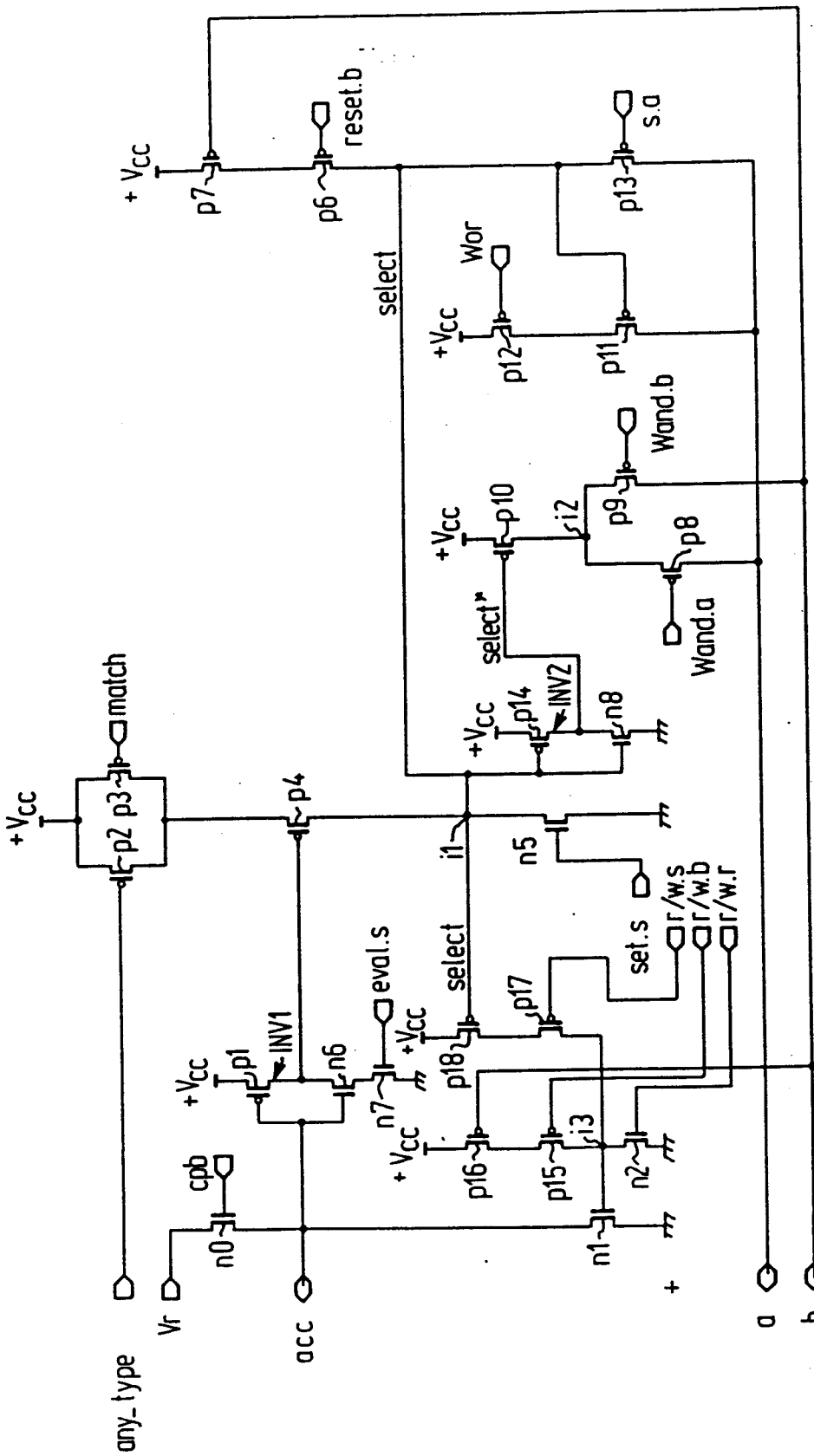
PIEŠ. 20



PIEŠ. 21B

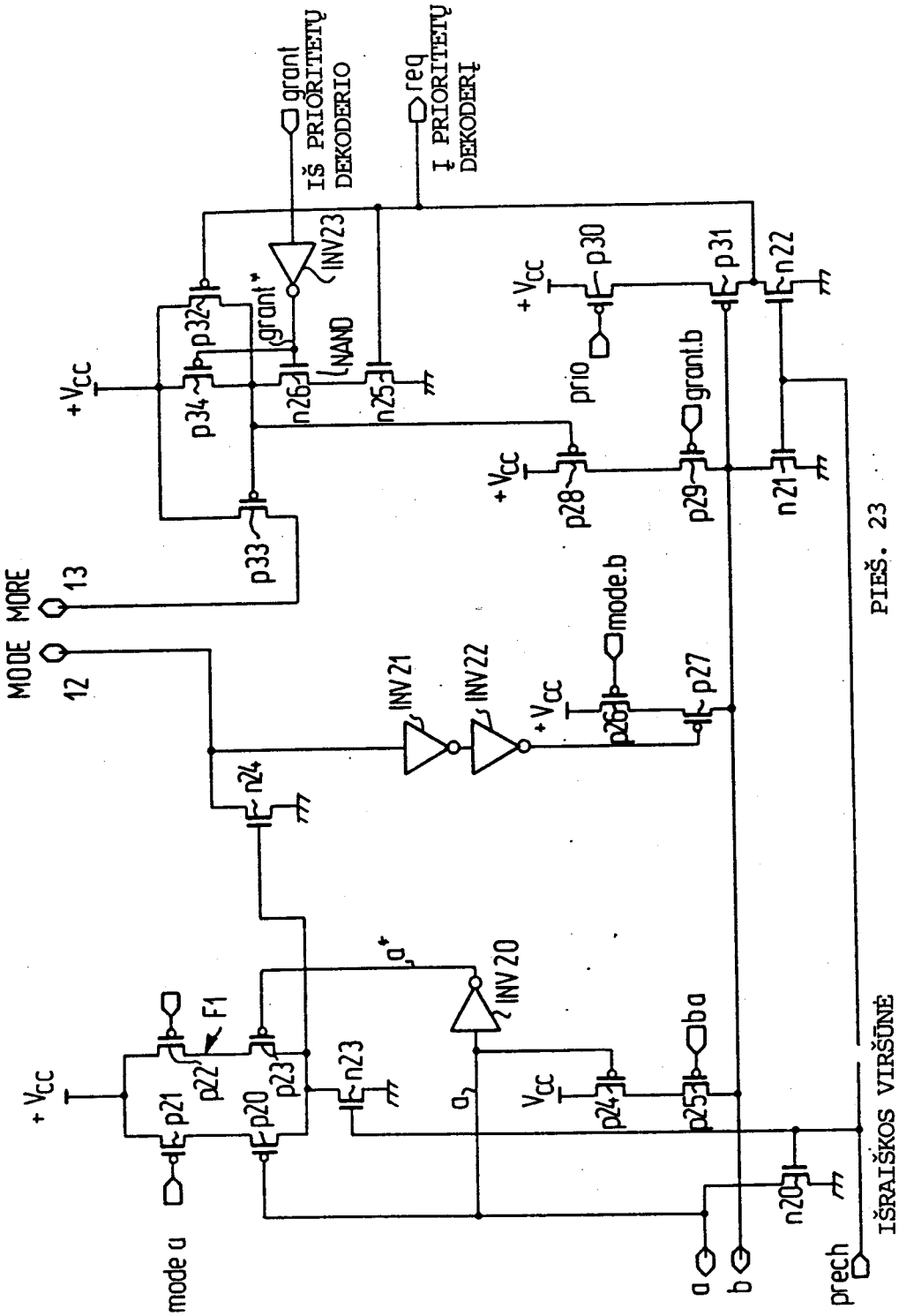


PIEŠ. 21 B



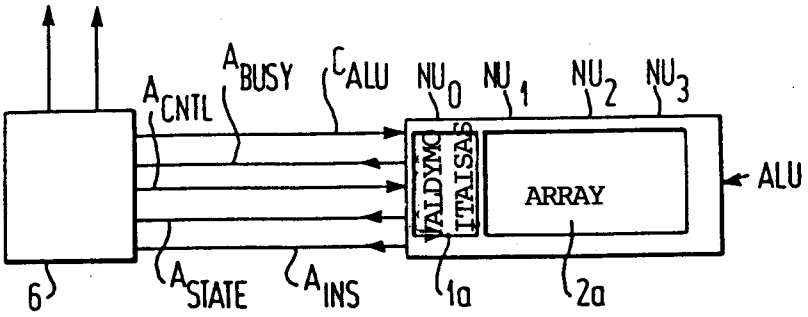
PIEŠ. 22

ELEMENTO VIRŠŪNĖ

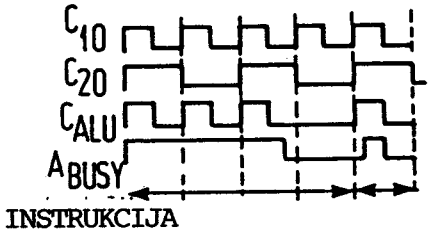

grant
IŠ PRIORITETŲ DEKODERIO
req
Į PRIORITETŲ DEKODERĮ
INV23
NAND
grant*
p32
p34
n26
n25
+Vcc
p33
MODE MORE
13
MODE
12
INV21
INV22
n24
p30
p31
n22
prio
grant.b
p28
p29
n21
mode.b
p26
p27
INV20
a*
F1
p21
p22
p20
p23
n23
a2
p24
p25
ba
n20
mode a
a
b
prech
IŠRAIŠKOS VIRŠŪNĖ

PIEŠ. 23

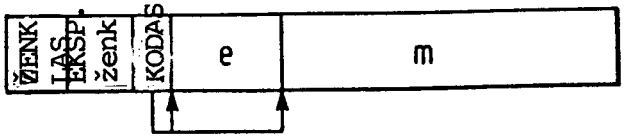
Į KITUS ĮRENGINIUS



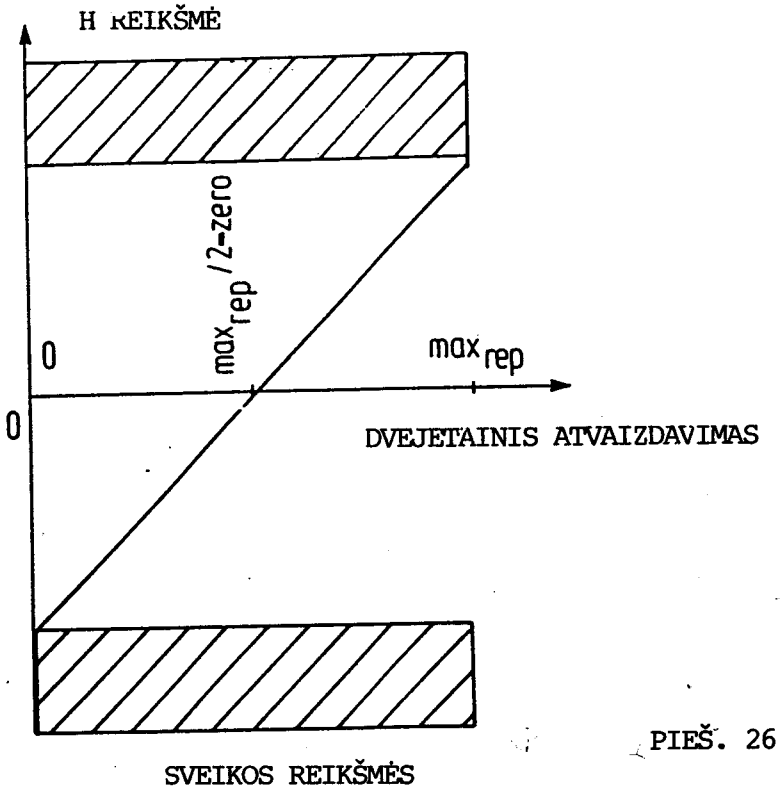
PIEŠ. 24 A



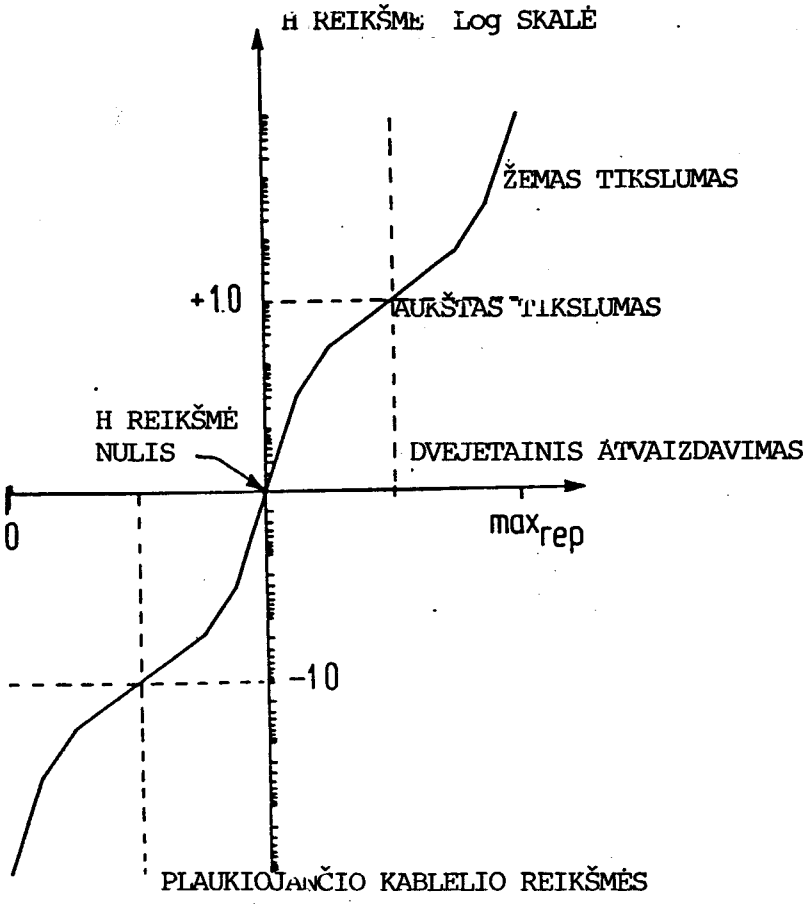
PIEŠ. 24 B



PIEŠ. 25

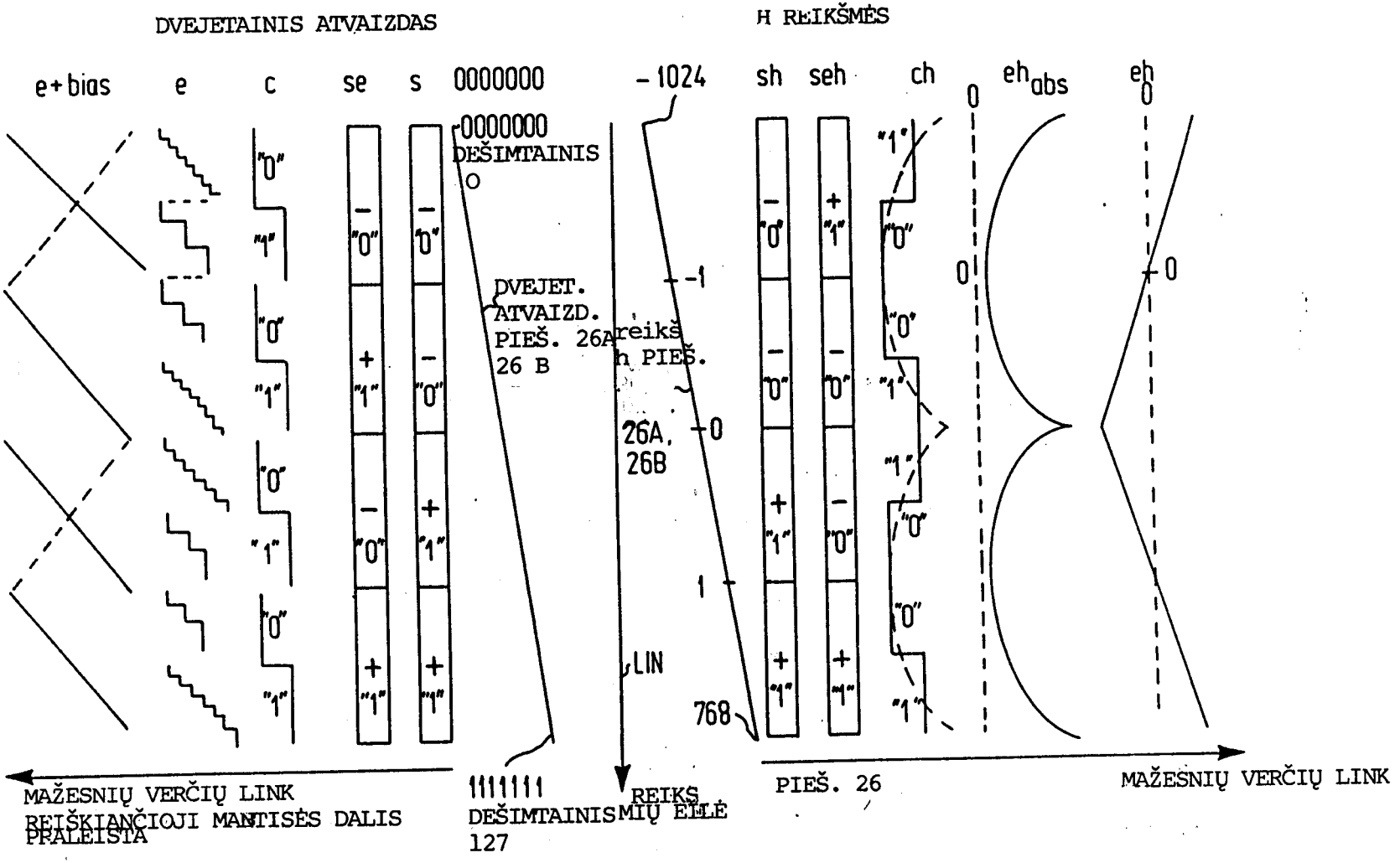


PIEŠ. 26 A



PIEŠ. 26 B

PLAUKIOJANČIO KABLELIO REIKŠMĖS



ĮPRASTINIS BINARINIS ATSPINDĖJIMAS		DEŠIMTAINIS BINARINIO DYDŽIO ATSPINDĖJIMAS											
			s,sh,	se,	seh,	c,	ch,	e,	e,	eh,	m,	m_d,	m_h,	val_h,
0000000	0	->	0	0,1	0,1	0,		9	000,1001		0,000		-2	-1024
0000001	1	->	0	0,1	0,1	0,		9	000,1001		1,001		-1.5	-768
0000010	2	->	0	0,1	0,1	1,		8	001,1000		0,000		-2	-512
0000011	3	->	0	0,1	0,1	1,		8	001,1000		1,001		-1.5	-384
0000100	4	->	0	0,1	0,1	2,		7	010,0111		0,000		-2	-256
0000101	5	->	0	0,1	0,1	2,		7	010,0111		1,001		-1.5	-192
0000110	6	->	0	0,1	0,1	3,		6	011,0110		0,000		-2	-128
0000111	7	->	0	0,1	0,1	3,		6	011,0110		1,001		-1.5	-96
0001000	8	->	0	0,1	0,1	4,		5	100,0101		0,000		-2	-64
0001001	9	->	0	0,1	0,1	4,		5	100,0101		1,001		-1.5	-48
0001010	10	->	0	0,1	0,1	5,		4	101,0100		0,000		-2	-32
0001011	11	->	0	0,1	0,1	5,		4	101,0100		1,001		-1.5	-24
0001100	12	->	0	0,1	0,1	6,		3	110,0011		0,000		-2	-16
0001101	13	->	0	0,1	0,1	6,		3	110,0011		1,001		-1.5	-12
0001110	14	->	0	0,1	0,1	7,		2	111,0010		0,000		-2	-8
0001111	15	->	0	0,1	0,1	7,		2	111,0010		1,001		-1.5	-6
0010000	16	->	0	0,1	1,0	0,		1	000,0001		0,000		-2	-4
0010001	17	->	0	0,1	1,0	0,		1	000,0001		1,001		-1.875	-3.75
0010010	18	->	0	0,1	1,0	0,		1	000,0001		2,010		-1.75	-3.5
0010011	19	->	0	0,1	1,0	0,		1	000,0001		3,011		-1.625	-3.25
0010100	20	->	0	0,1	1,0	0,		1	000,0001		4,100		-1.5	-3
0010101	21	->	0	0,1	1,0	0,		1	000,0001		5,101		-1.375	-2.75
0010110	22	->	0	0,1	1,0	0,		1	000,0001		6,110		-1.25	-2.5
0010111	23	->	0	0,1	1,0	0,		1	000,0001		7,111		-1.125	-2.25
0011000	24	->	0	0,1	1,0	1,		0	001,0000		0,000		-2	-2
0011001	25	->	0	0,1	1,0	1,		0	001,0000		1,001		-1.875	-1.875
0011010	26	->	0	0,1	1,0	1,		0	001,0000		2,010		-1.75	-1.75
0011011	27	->	0	0,1	1,0	1,		0	001,0000		3,011		-1.625	-1.625
0011100	28	->	0	0,1	1,0	1,		0	001,0000		4,100		-1.5	-1.5
0011101	29	->	0	0,1	1,0	1,		0	001,0000		5,101		-1.375	-1.375
0011110	30	->	0	0,1	1,0	1,		0	001,0000		6,110		-1.25	-1.25
0011111	31	->	0	0,1	1,0	1,		0	001,0000		7,111		-1.125	-1.125

PIEŠ. 27A

ĮPRASTINIS BINARINIS ATSPINDĖJIMAS	DEŠIMTAINIS	BINARINIO DYDŽIO ATSPINDĖJIMAS s,sh,	se,	seh,	c,	ch,	e,	eh,	e,	eh,	m_d,	$m_?$	$m_{h?}$	$val_{h?}$
0100000	32 ->	0	1,0	0,0	0,		-1	000,0000	0,000			-2		-1
0100001	33 ->	0	1,0	0,0	0,		-1	000,0000	1,001		-1.875		-.9375	
0100010	34 ->	0	1,0	0,0	0,		-1	000,0000	2,010		-1.75		-.875	
0100011	35 ->	0	1,0	0,0	0,		-1	000,0000	3,011		-1.625		-.8125	
0100100	36 ->	0	1,0	0,0	0,		-1	000,0000	4,100		-1.5		-.75	
0100101	37 ->	0	1,0	0,0	0,		-1	000,0000	5,101		-1.375		-.6875	
0100110	38 ->	0	1,0	0,0	0,		-1	000,0000	6,110		-1.25		-.625	
0100111	39 ->	0	1,0	0,0	0,		-1	000,0000	7,111		-1.125		-.5625	
0101000	40 ->	0	1,0	0,0	1,		-2	001,0000	0,000			-2		-.5
0101001	41 ->	0	1,0	0,0	1,		-2	001,0000	1,001		-1.875		-.46875	
0101010	42 ->	0	1,0	0,0	1,		-2	001,0000	2,010		-1.75		-.4375	
0101011	43 ->	0	1,0	0,0	1,		-2	001,0000	3,011		-1.625		-.40625	
0101100	44 ->	0	1,0	0,0	1,		-2	001,0000	4,100		-1.5		-.375	
0101101	45 ->	0	1,0	0,0	1,		-2	001,0000	5,101		-1.375		-.34375	
0101110	46 ->	0	1,0	0,0	1,		-2	001,0000	6,110		-1.25		-.3125	
0101111	47 ->	0	1,0	0,0	1,		-2	001,0000	7,111		-1.125		-.28125	
0110000	48 ->	0	1,0	1,1	0,		-3	000,0000	0,000			-2		-.25
0110001	49 ->	0	1,0	1,1	0,		-3	000,0000	1,001		-1.5		-.1875	
0110010	50 ->	0	1,0	1,1	1,		-4	001,0000	0,000			-2		-.125
0110011	51 ->	0	1,0	1,1	1,		-4	001,0000	1,001		-1.5		-0.09375	
0110100	52 ->	0	1,0	1,1	2,		-5	010,0000	0,000			-2		-0.0625
0110101	53 ->	0	1,0	1,1	2,		-5	010,0000	1,001		-1.5		-0.046875	
0110110	54 ->	0	1,0	1,1	3,		-6	011,0000	0,000			-2		-0.03125
0110111	55 ->	0	1,0	1,1	3,		-6	011,0000	1,001		-1.5		-0.0234375	
0111000	56 ->	0	1,0	1,1	4,		-7	100,0000	0,000			-2		-0.015625
0101001	57 ->	0	1,0	1,1	4,		-7	100,0000	1,001		-1.5		-0.0117188	
0111010	58 ->	0	1,0	1,1	5,		-8	101,0000	0,000			-2		-0.0078125
0111011	59 ->	0	1,0	1,1	5,		-8	101,0000	1,001		-1.5		-0.00585938	
0111100	60 ->	0	1,0	1,1	6,		-9	110,0000	0,000			-2		-0.00390625
0111101	61 ->	0	1,0	1,1	6,		-9	110,0000	1,001		-1.5		-0.00292969	
0111110	62 ->	0	1,0	1,1	7,		-10	111,0000	0,000			-2		-0.00195312
0111111	63 ->	0	1,0	1,1	7,		-10	111,0000	1,001		-1.5		-0.00146484	

24/37

PIEŠ. 27B

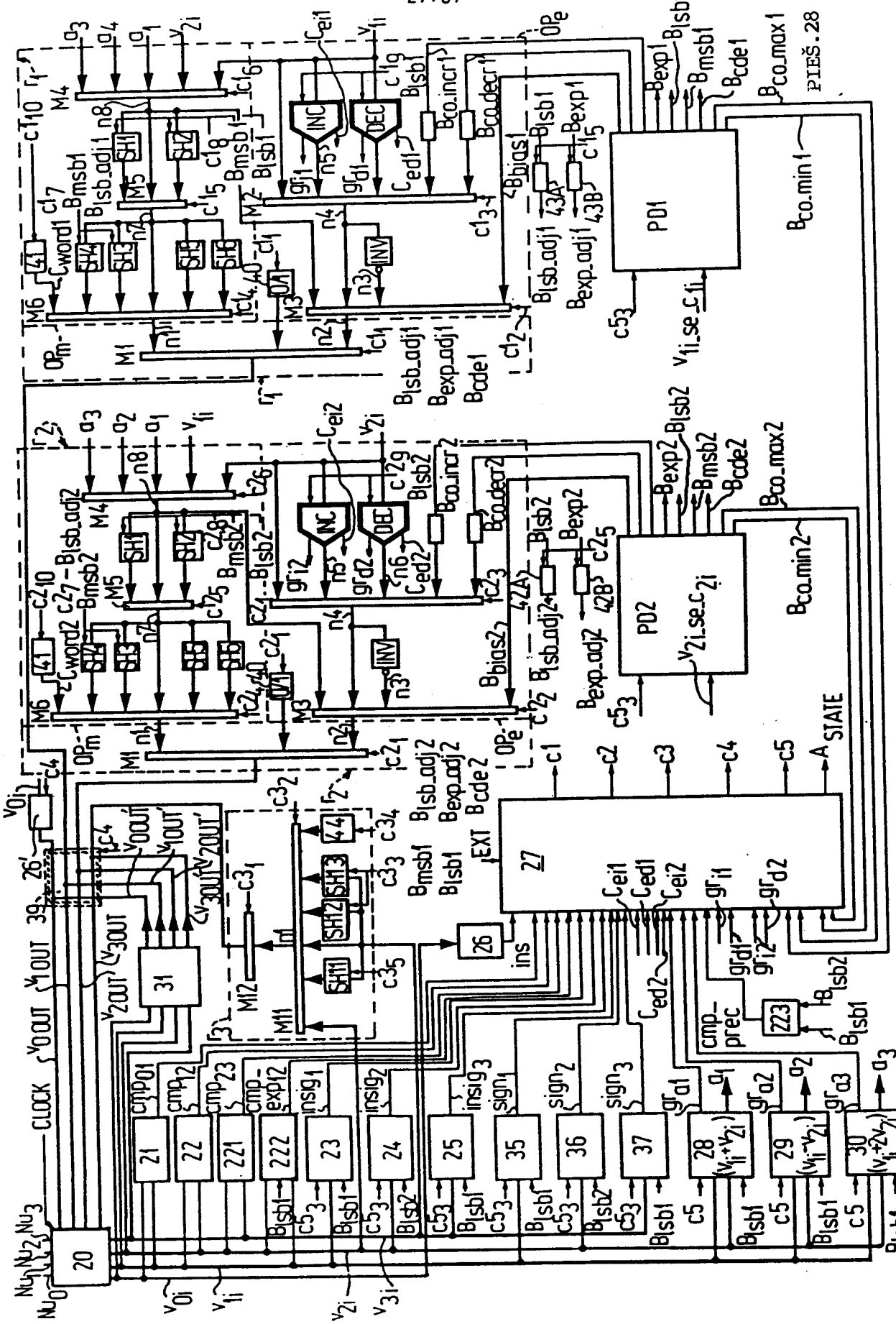
192

ĮPRASTINIS BINARINIS ATSPINDĖJIMAS			DEŠIMTAINIS BINARINIO DYDŽIO ATSPINDĖJIMAS s,sh,	se,	seh,	c,	ch,	e,	eh,	e,	eh,	m,	m_d,	m_h,	val_h,
															0
1000000	64	->	1	0,0	0,1	0,	-10	000,0000		0,000		1			
1000001	65	->	1	0,0	0,1	0,	-10	000,0000		1,001		1.5			0.00146484
1000010	66	->	1	0,0	0,1	1,	-9	001,0000		0,000		1			0.00195312
1000011	67	->	1	0,0	0,1	1,	-9	001,0000		1,001		1.5			0.00292969
1000100	68	->	1	0,0	0,1	2,	-8	010,0000		0,000		1			0.00390625
1000101	69	->	1	0,0	0,1	2,	-8	010,0000		1,001		1.5			0.00585938
1000110	70	->	1	0,0	0,1	3,	-7	011,0000		0,000		1			0.0078125
1000111	71	->	1	0,0	0,1	3,	-7	011,0000		1,001		1.5			0.0117188
1001000	72	->	1	0,0	0,1	4,	-6	100,0000		0,000		1			0.015625
1001001	73	->	1	0,0	0,1	4,	-6	100,0000		1,001		1.5			0.0234375
1001010	74	->	1	0,0	0,1	5,	-5	101,0000		0,000		1			0.03125
1001011	75	->	1	0,0	0,1	5,	-5	101,0000		1,001		1.5			0.046875
1001100	76	->	1	0,0	0,1	6,	-4	110,0000		0,000		1			0.0625
1001101	77	->	1	0,0	0,1	6,	-4	110,0000		1,001		1.5			0.09375
1001110	78	->	1	0,0	0,1	7,	-3	111,0000		0,000		1			.125
1001111	79	->	1	0,0	0,1	7,	-3	111,0000		1,001		1.5			.1875
1010000	80	->	1	0,0	1,0	0,	-2	000,0000		0,000		1			.25
1010001	81	->	1	0,0	1,0	0,	-2	000,0000		1,001		1.125			.28125
1010010	82	->	1	0,0	1,0	0,	-2	000,0000		2,010		1.25			.3125
1010011	83	->	1	0,0	1,0	0,	-2	000,0000		3,011		1.375			.34375
1010100	84	->	1	0,0	1,0	0,	-2	000,0000		4,100		1.5			.375
1010101	85	->	1	0,0	1,0	0,	-2	000,0000		5,101		1.625			.40625
1010110	86	->	1	0,0	1,0	0,	-2	000,0000		6,110		1.75			.4375
1010111	87	->	1	0,0	1,0	0,	-2	000,0000		7,111		1.875			.46875
1011000	88	->	1	0,0	1,0	1,	-1	001,0000		0,000		1			.5
1011001	89	->	1	0,0	1,0	1,	-1	001,0000		1,001		1.125			.5625
1011010	90	->	1	0,0	1,0	1,	-1	001,0000		2,010		1.25			.625
1011011	91	->	1	0,0	1,0	1,	-1	001,0000		3,011		1.375			.6875
1011100	92	->	1	0,0	1,0	1,	-1	001,0000		4,100		1.5			.75
1011101	93	->	1	0,0	1,0	1,	-1	001,0000		5,101		1.625			.8125
1011110	94	->	1	0,0	1,0	1,	-1	001,0000		6,110		1.75			.875
1011111	95	->	1	0,0	1,0	1,	-1	001,0000		7,111		1.875			.9375

PIEŠ. 27c

ĮPRASTĖNIS BINARINIS ATSPINDĖJIMAS	DEŠIMTAINIS BINARINIO DYDŽIO ATSPINDĖJIMAS													
			s	sh	se	seh	c	ch	e	eh	e	eh	m	m_d	m_h	val_h
1100000	96	->	1	1,1	0,0	0	0,	0	000,0000	0,000			1		1	
1100001	97	->	1	1,1	0,0	0	0,	0	000,0000	1,001	1.125		1.125		1.125	
1100010	98	->	1	1,1	0,0	0	0,	0	000,0000	2,010	1.25		1.25		1.25	
1100011	99	->	1	1,1	0,0	0	0,	0	000,0000	3,011	1.375		1.375		1.375	
1100100	100	->	1	1,1	0,0	0	0,	0	000,0000	4,100	1.5		1.5		1.5	
1100101	101	->	1	1,1	0,0	0	0,	0	000,0000	5,101	1.625		1.625		1.625	
1100110	102	->	1	1,1	0,0	0	0,	0	000,0000	6,111	1.75		1.75		1.75	
1100111	103	->	1	1,1	0,0	0	0,	0	000,0000	7,111	1.875		1.875		1.875	
1101000	104	->	1	1,1	0,0	1	1,	1	001,0001	0,000			1		2	
1101001	105	->	1	1,1	0,0	1	1,	1	001,0001	1,001	1.125		2.25		2.25	
1101010	106	->	1	1,1	0,0	1	1,	1	001,0001	2,010	1.25		2.5		2.5	
1101011	107	->	1	1,1	0,0	1	1,	1	001,0001	3,011	1.375		2.75		2.75	
1101100	108	->	1	1,1	0,0	1	1,	1	001,0001	4,100	1.5		3		3	
1101101	109	->	1	1,1	0,0	1	1,	1	001,0001	5,101	1.625		3.25		3.25	
1101110	110	->	1	1,1	0,0	1	1,	1	001,0001	6,110	1.75		3.5		3.5	
1101111	111	->	1	1,1	0,0	1	1,	1	001,0001	7,111	1.875		3.75		3.75	
1110000	112	->	1	1,1	1,1	0	2,	2	000,0010	0,000			1		4	
1110001	113	->	1	1,1	1,1	0	2,	2	000,0010	1,001	1.5		6		6	
1110010	114	->	1	1,1	1,1	1	3,	3	001,0011	0,000			1		8	
1110011	115	->	1	1,1	1,1	1	3,	3	001,0011	1,001	1.5		12		12	
1110100	116	->	1	1,1	1,1	2	4,	4	010,0100	0,000			1		16	
1110101	117	->	1	1,1	1,1	2	4,	4	010,0100	1,001	1.5		24		24	
1110110	118	->	1	1,1	1,1	3	5,	5	011,0101	0,000			1		32	
1110111	119	->	1	1,1	1,1	3	5,	5	011,0101	1,001	1.5		48		48	
1111000	120	->	1	1,1	1,1	4	6,	6	100,0110	0,000			1		64	
1111001	121	->	1	1,1	1,1	4	6,	6	100,0110	1,001	1.5		96		96	
1111010	122	->	1	1,1	1,1	5	7,	7	101,0111	0,000			1		128	
1111011	123	->	1	1,1	1,1	5	7,	7	101,0111	1,001	1.5		192		192	
1111100	124	->	1	1,1	1,1	6	8,	8	110,1000	0,000			1		256	
1111101	125	->	1	1,1	1,1	6	8,	8	110,1000	1,001	1.5		384		384	
1111110	126	->	1	1,1	1,1	7	9,	9	111,1001	0,000			1		512	
1111111	127	->	1	1,1	1,1	7	9,	9	111,1001	1,001	1.5		768			

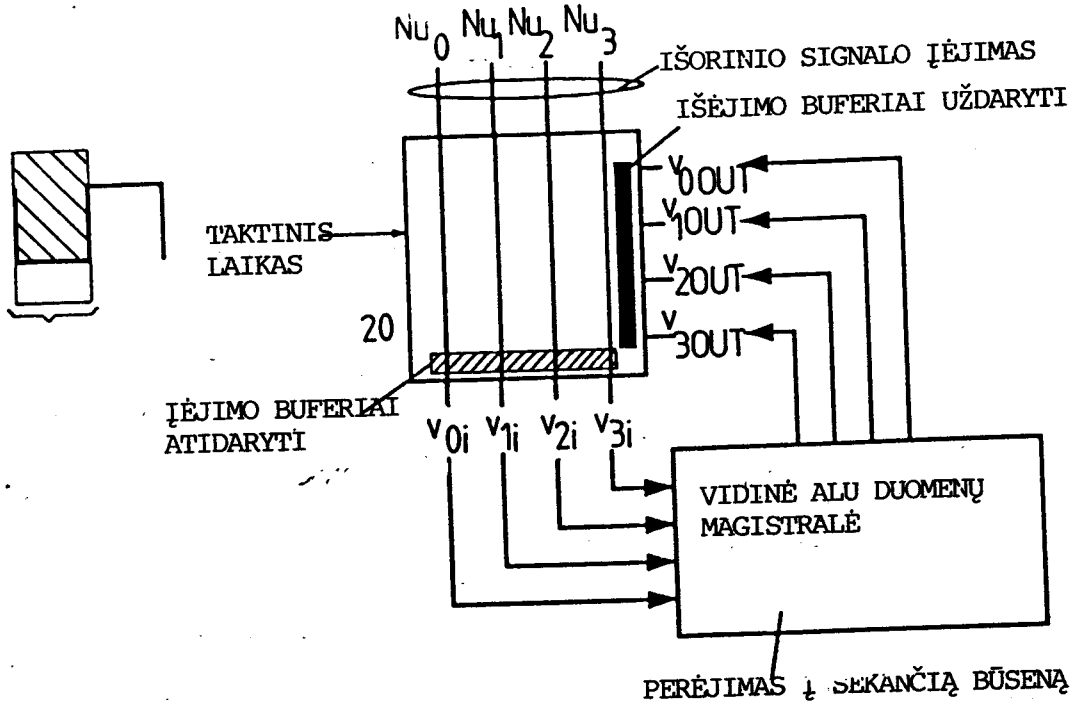
PIEŠ. 27 D



PIEŠ. 28

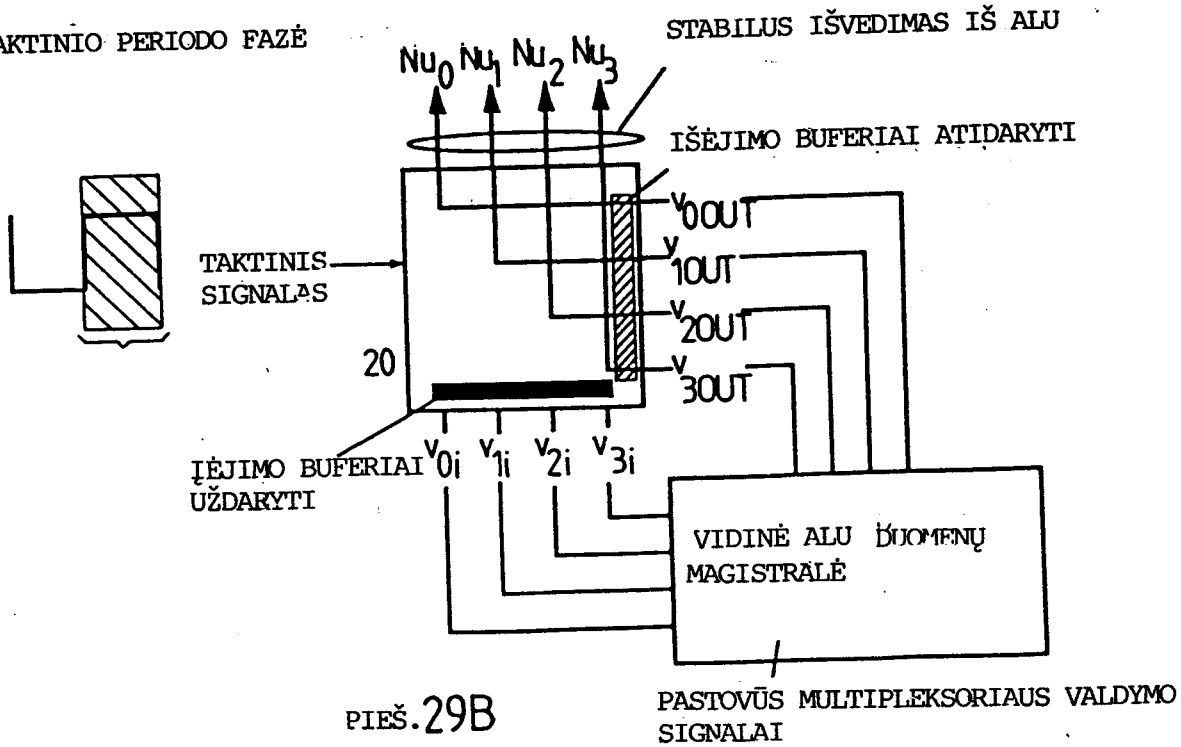
ĮĖJIMO/IŠĖJIMO BUFERIO 20 TAKTAVIMAS

INSTRUKCIJOS TAKTINIO PERIODO PIRMOJI O-FAZĖ



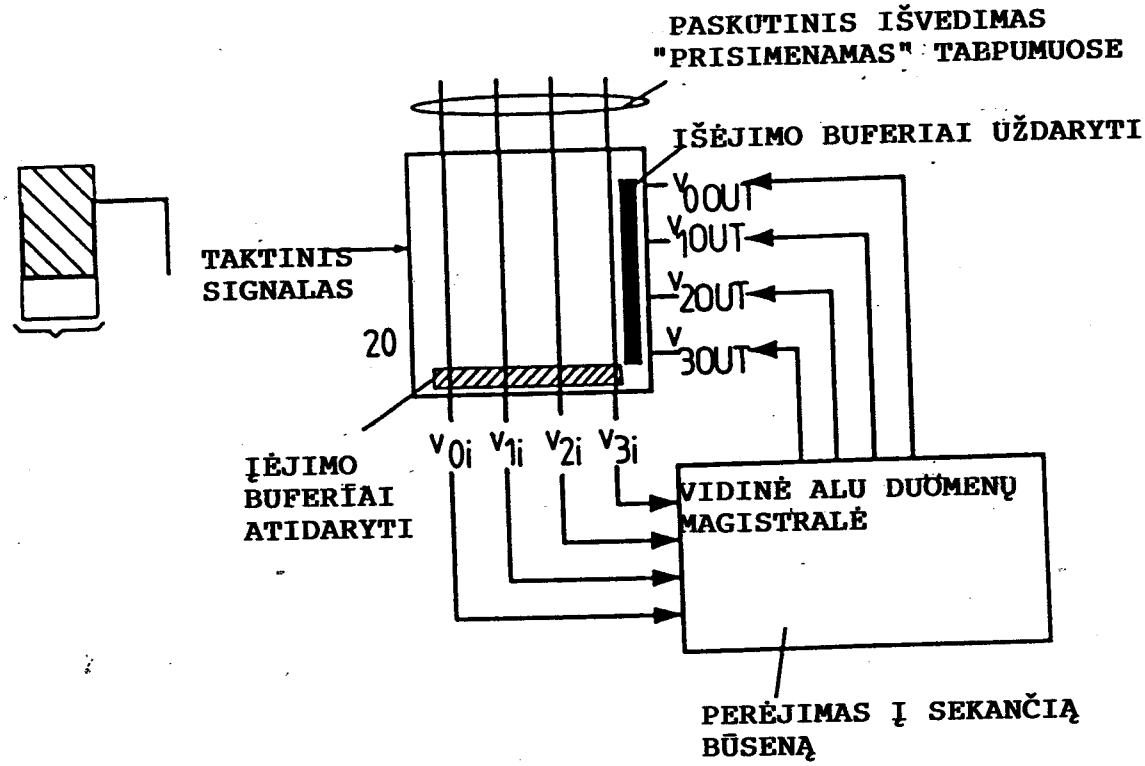
PIEŠ. 29A

1 TAKTINIO PERIODO FAZĖ

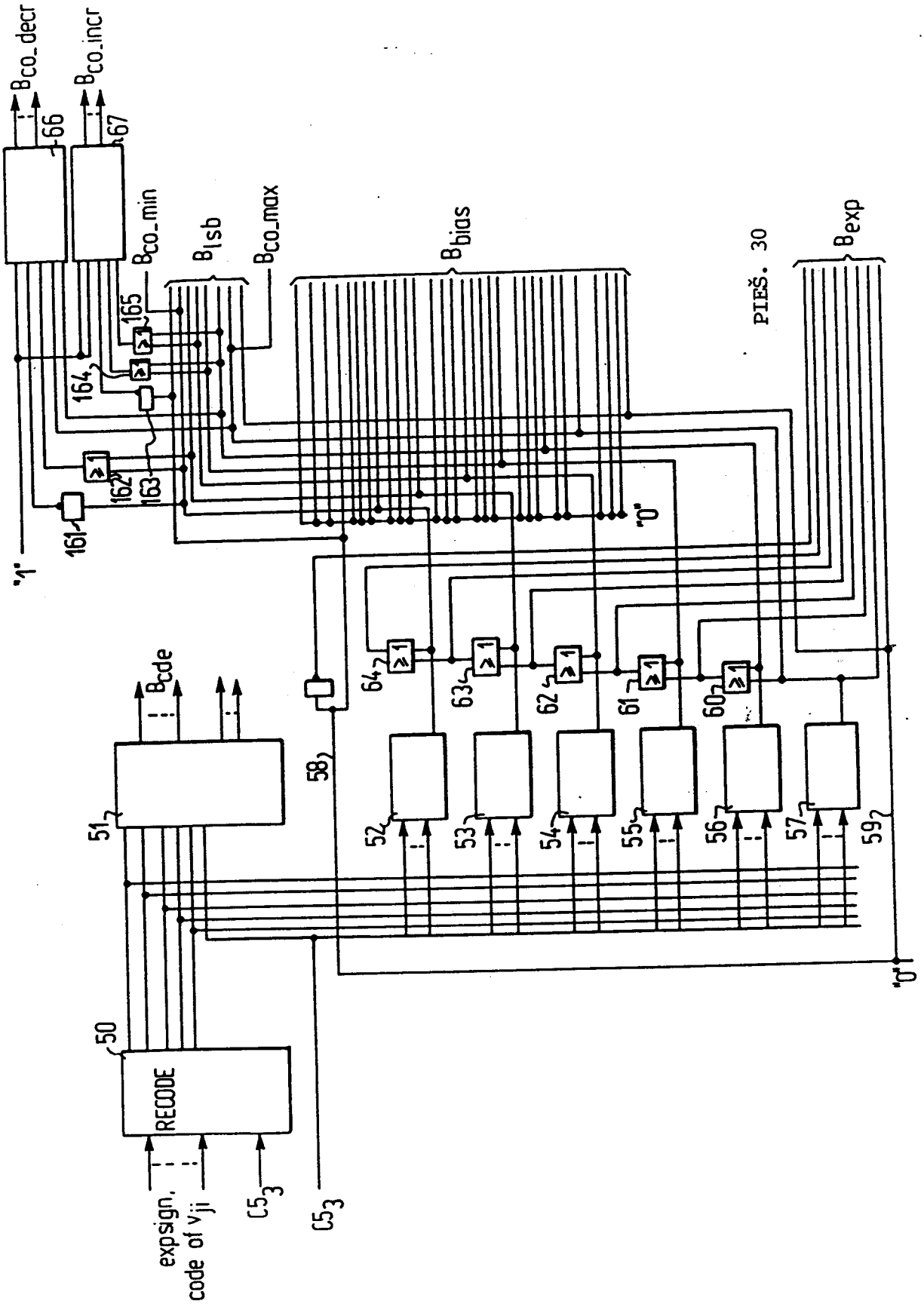


PIEŠ. 29B

DAUGIACIKLĖS INSTRUKCIJOS TAKTINIO PERIODO 0 FAZĖS DALIS

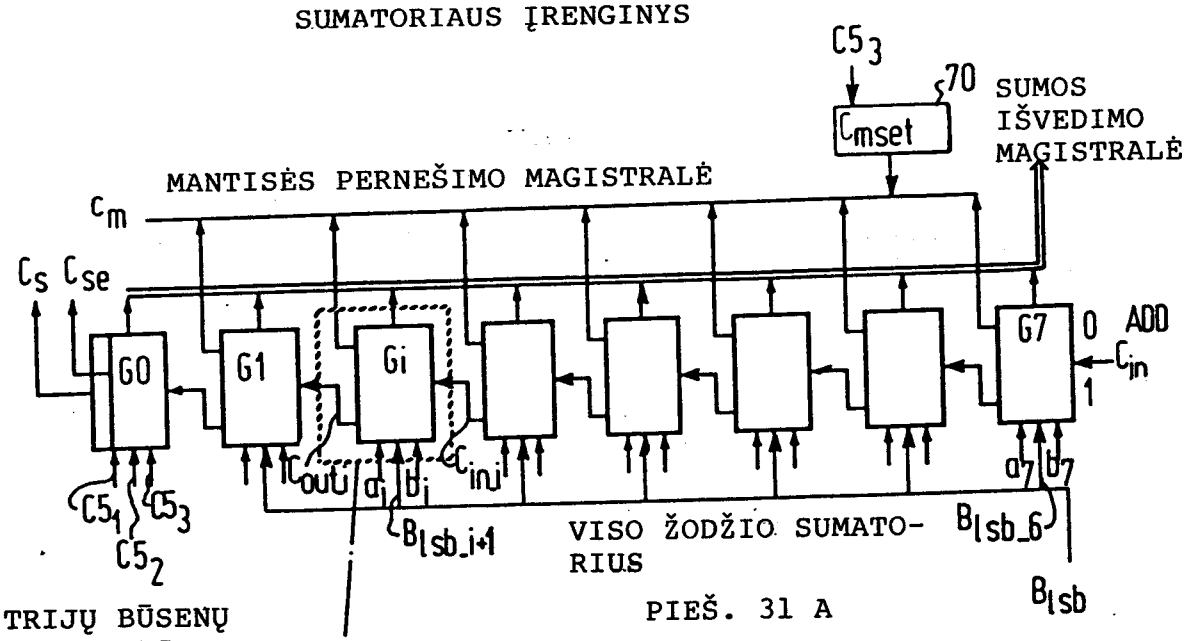


PIEŠ. 29 C

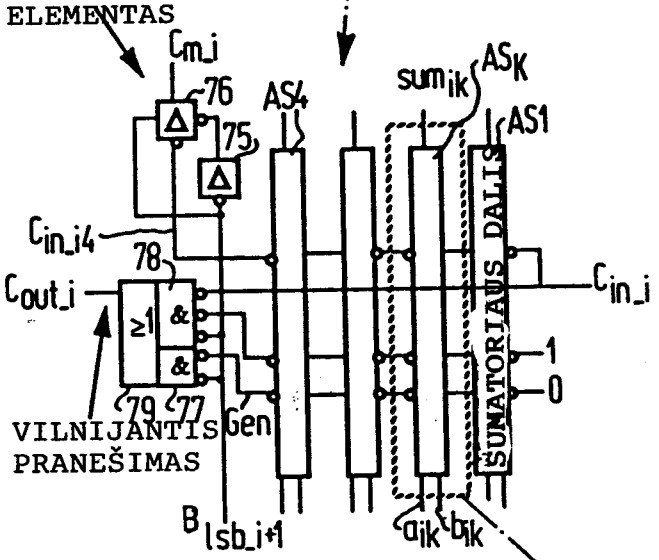

Bco_decr
Bco_incr
66
67
Bco_min
Blsb
Bco_max
Bbias
Bexp
165
164
162
163
161
"1"
"0"
Bcde
51
58
64
63
62
61
60
52
53
54
55
56
57
59
50
RECODE
exp sign, code of vji
C5 3
C5 3
"0"

PIEŠ. 30

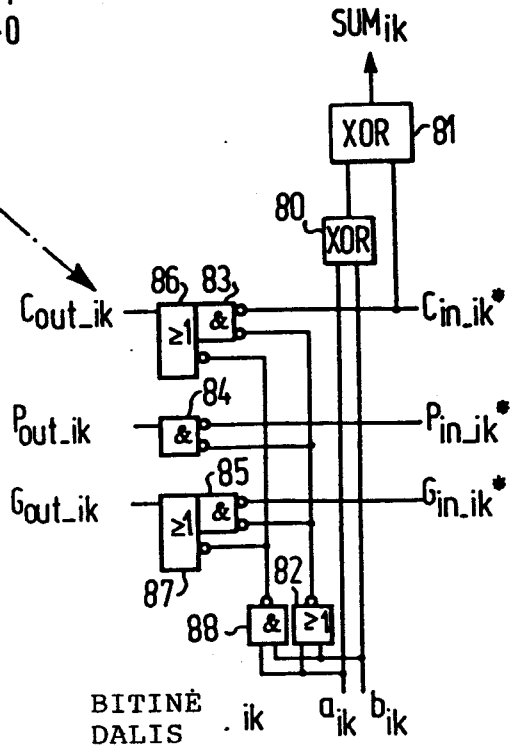
SUMATORIAUS ĮRENGINYS



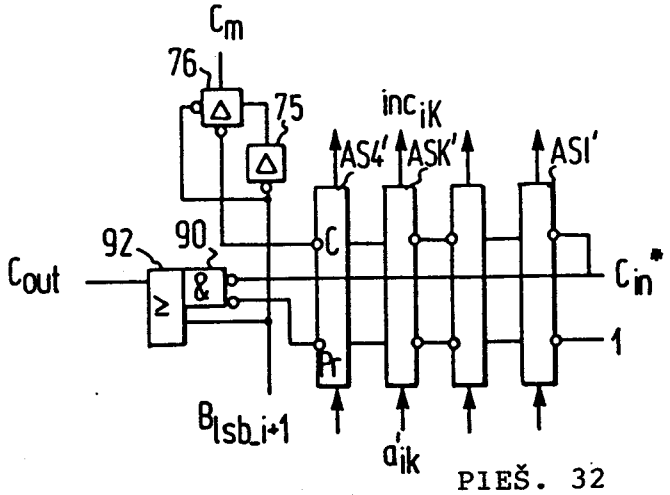
PIEŠ. 31 A



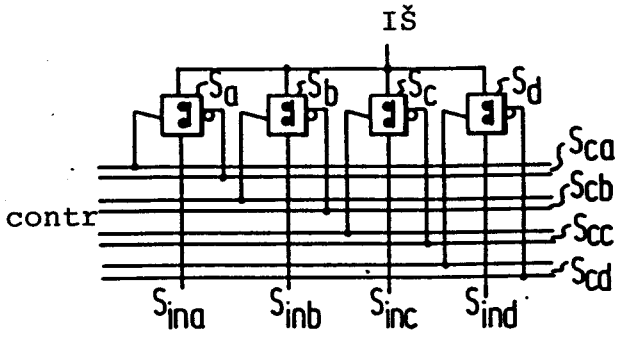
PIEŠ. 31 B



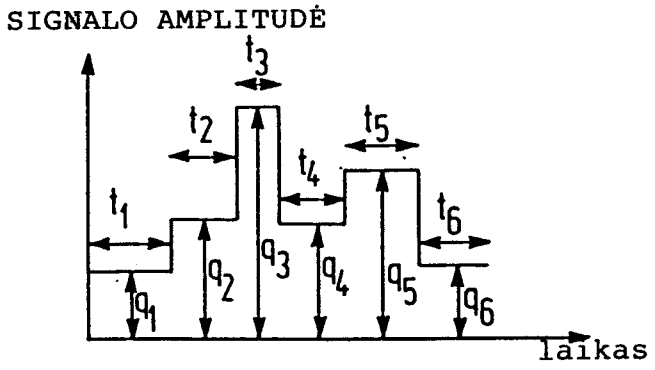
PIEŠ. 31 C



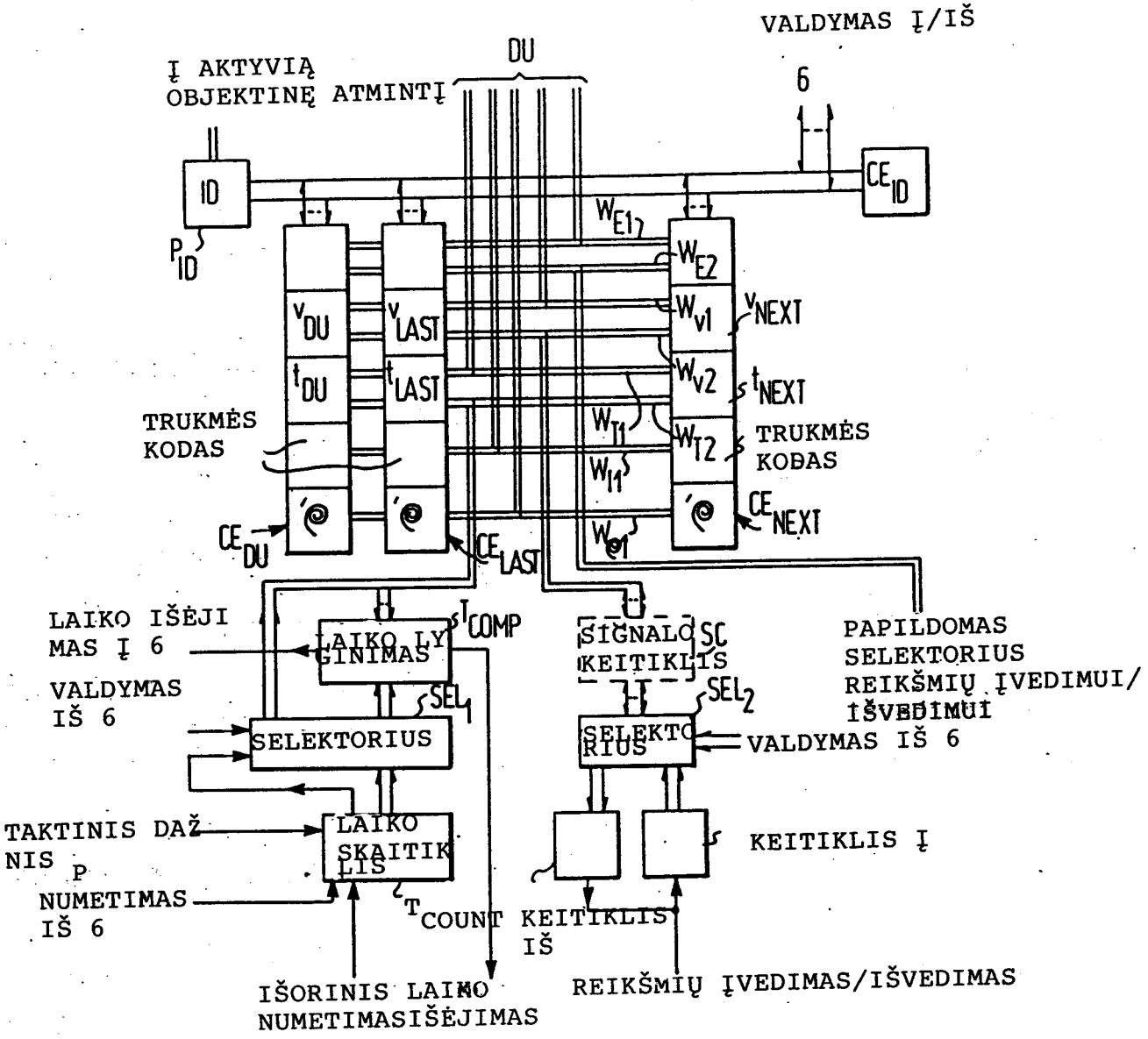
PIEŠ. 32



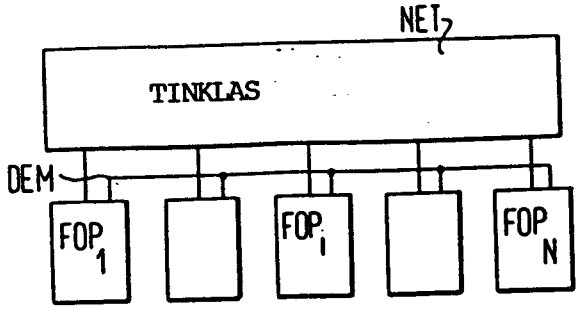
PIEŠ?33



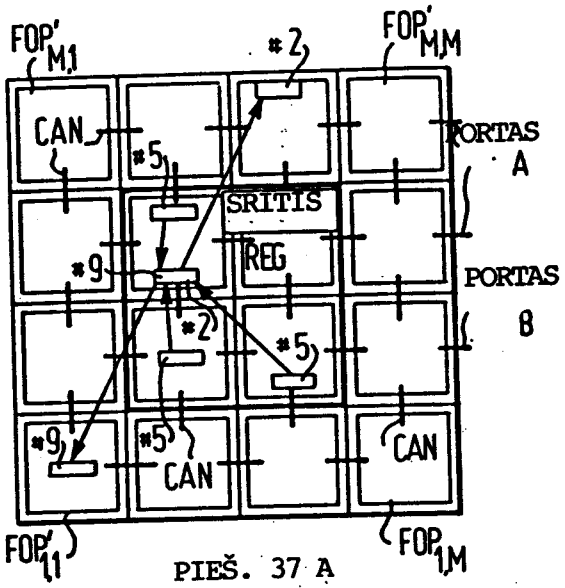
PIEŠ. 34


VALDYMAS Į/IŠ
Į AKTYVIĄ OBJEKTINĘ ATMINTĮ
DU
6
ID
CE_ID
P_ID
W_E1
W_E2
W_v1
W_v2
W_T1
W_T2
W_11
W_01
v_DU
v_LAST
v_NEXT
t_DU
t_LAST
t_NEXT
TRUKMĖS KODAS
TRUKMĖS KODAS
CE_DU
CE_LAST
CE_NEXT
LAIKO IŠĖJIMAS Į 6
LAIKO LYGINIMAS
T_COMP
SIGNALO KEITIKLIS
SC
PAPILDOMAS SELEKTORIUS REIKŠMIŲ ĮVEDIMUI/IŠVEDIMUI
VALDYMAS IŠ 6
SEL_1
SEL_2
SELEKTORIUS
SELEKTORIUS
VALDYMAS IŠ 6
TAKTINIS DAŽNIS P
LAIKO SKAITIKLIS
KEITIKLIS Į
NUMETIMAS IŠ 6
T_COUNT
KEITIKLIS IŠ
IŠORINIS LAIKO NUMETIMAS
IŠĖJIMAS
REIKŠMIŲ ĮVEDIMAS/IŠVEDIMAS

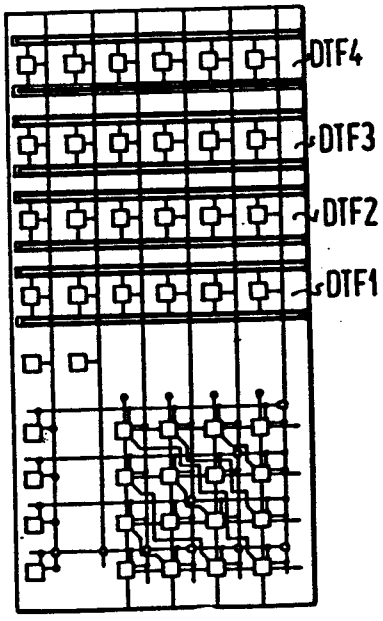
PIEŠ.3



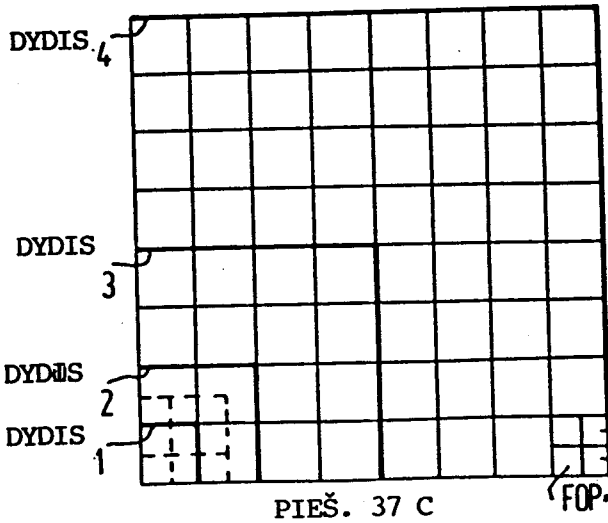
PIEŠ. 36



PIEŠ. 37 A



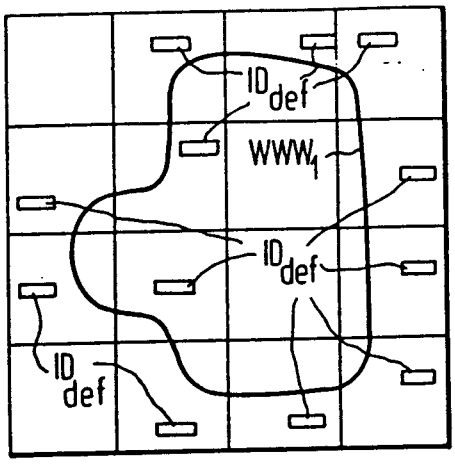
PIEŠ. 37 B



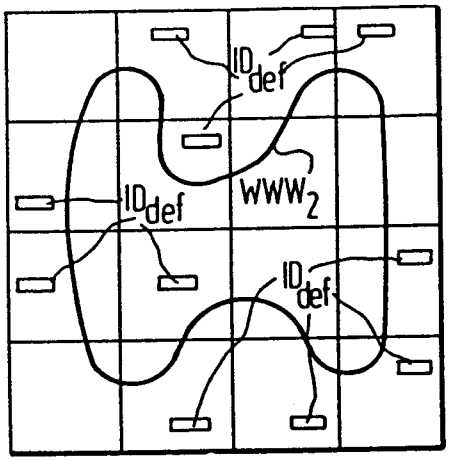
PIEŠ. 37 C

ANTROS EILĖS PROCESORIAUS, SUSIDEDANČIO IŠ 256FOP'ų DALINIMAS Į SRITIS:

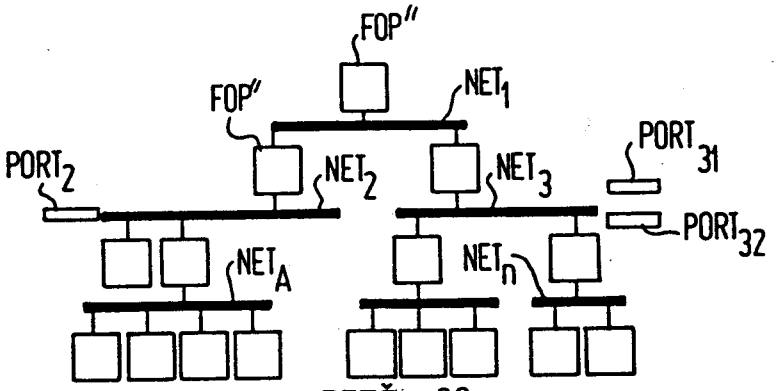
256 SRITYS O DYDIS
64 SRITYS 1 DYDIS
16 SRITYS 2 DYDIS
4 SRITYS 3 DYDIS
1 SRITIS 4 DYDIS



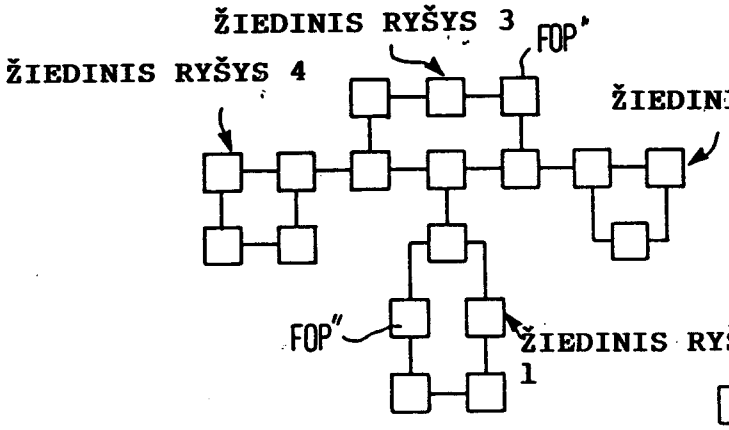
PIEŠ. 37 D



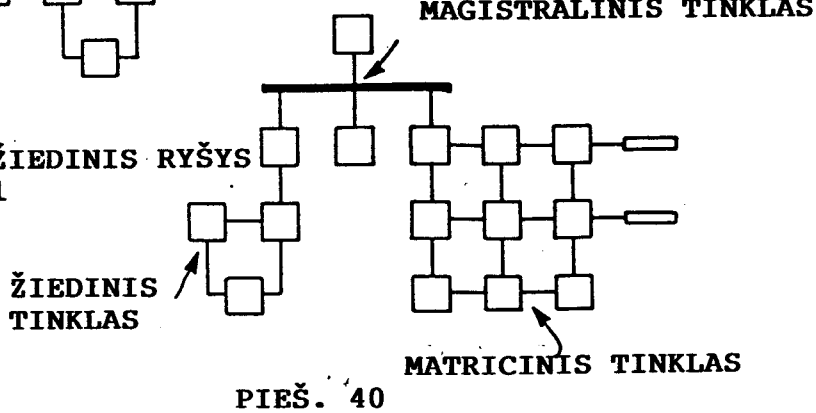
PIEŠ. 37 E



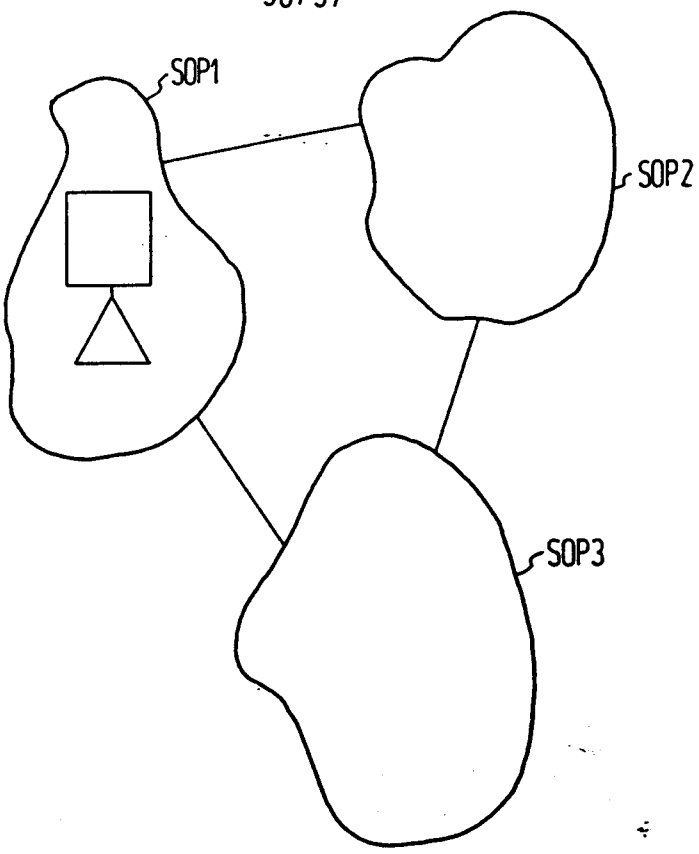
PIEŠ. 38



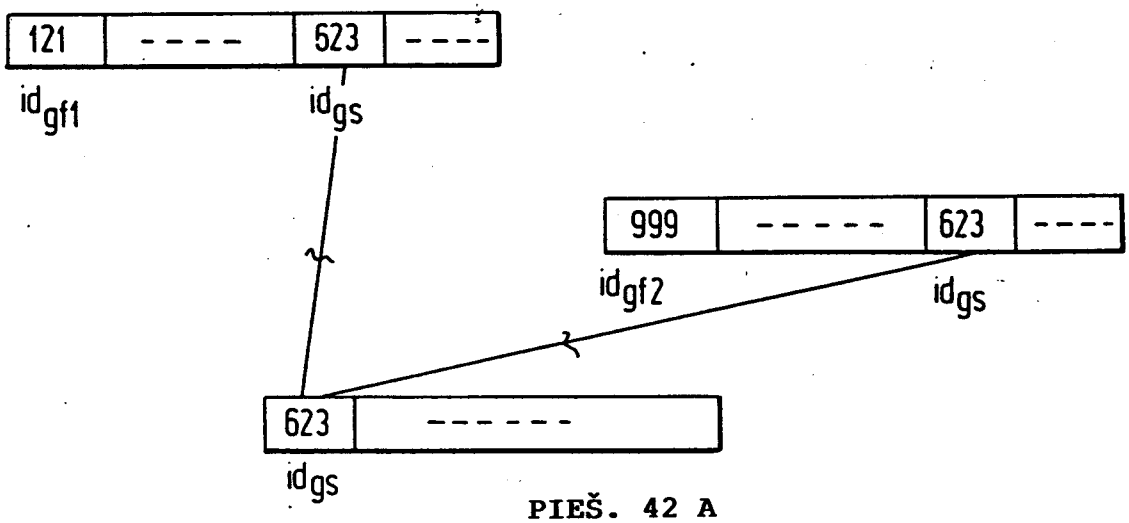
PIEŠ. 39



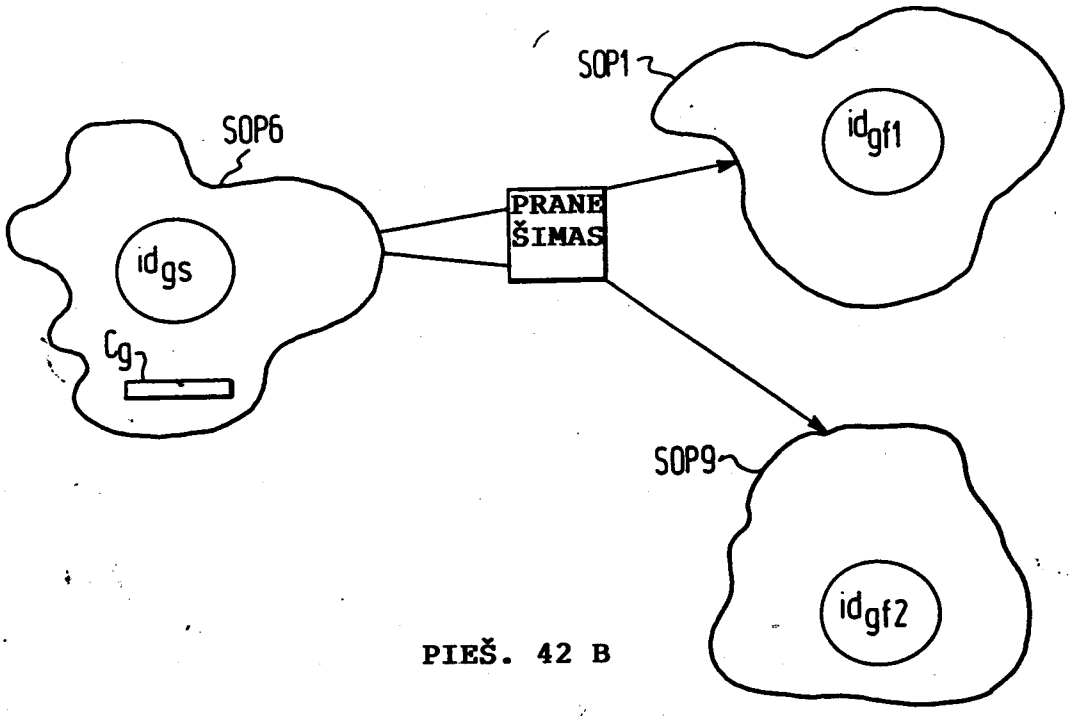
PIEŠ. 40



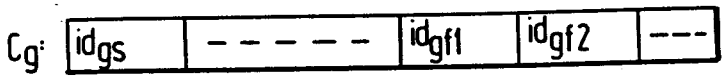
PIEŠ. 41



PIEŠ. 42 A



PIEŠ. 42 B



PIEŠ. 42C