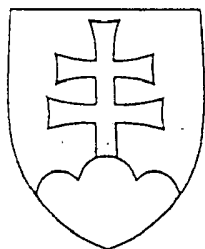


SLOVENSKÁ
REPUBLIKA

(19)



ÚRAD
PRIEMYSELNÉHO
VLASTNÍCTVA

ZVEREJNENÁ PRIHLÁŠKA VYNÁLEZU

(21) 4025-92

(13) A3

5(51) G 06 F 9/44, 15/82

(22) 30.12.92

(32) 02.08.90

(31) 9002558-6

(33) SE

(40) 08.09.93

(71) CARLSTEDT ELEKTRONIK AB, Partille, SE;

(72) CARSTEDT Lars Gunnar, Partille, SE;

(54) Redukčný procesor

(57) Redukčný procesor je riadený programom, ktorý má štruktúru a je prispôsobený na redukciu uvedenej štruktúry v niekoľkých redukčných krokoch vrátane rôznych druhov redukcie. Procesor praveho rádu tohto druhu obsahuje asociatívne aktívne ukladanie (1, 2), ktoré súčasne obsahuje množinu aktívnych pamäťových buniek (10, 2), z ktorých každá je schopná vyvolať vykonanie redukčnej operácie a komunikačnú sieť ($t_1, t_2, id, env, v_0, v_1, v_2, v_3, 12, 13, 14, 6, 7, 11, 16, 17$), ktorá odovzdáva výsledok každej redukcie asociatívne všetkým bunkám medzi pamäťovými bunkami, ktoré majú vzťah k uvedenému výsledku.

Redukčný procesor

Oblasť techniky

Tento vynález sa týka redukčného procesoru.

Doterajší stav techniky

Počítač bol vynájdený v priebehu 40. rokov. Odvtedy bol vyvíjaný revolučnou rýchlosťou. Napriek tomu dnešné počítače majú takmer rovnakú architektúru ako prvé počítače.

Najviac vylepšení bolo urobené v oblasti hardware. Uvedenie technológie VLSI a vylepšenie v litografii umožnilo postaviť jednočipové počítače, ktoré iba pred pár rokmi boli nazývané super počítače. Rozmery sa zmenšili exponenciálne a šírka spoja je teraz menšia ako 1 mikrometer. Taktovacia frekvencia a počet aktívnych tranzistorov sa zvýšili o niekoľko rádov. Fyzikálna hranica šírky spoja je pravdepodobne 0,2 mikrometra.

Zároveň sa ale architekti počítačov nezlepšili v používaní kremíka. Naopak, väčšina počítačov používala viac ako optimálne množstvo kremíka, aby boli rýchlejší.

Obe tieto skutočnosti zastavili vývoj rýchlosti samotných procesorov na nasledujúcich päť rokov. Sú uvedené na trh paralelné procesory, ktoré znamenajú vyššie náklady na hardware vzhľadom ku zvyšujúcej sa zložitosti a pre väčšinu programov neúmerné náklady na programovanie.

Náklady na hardware sa teda znížili ale náklady na programovanie nových systémov sa výrazne zvýšili a rýchlo sa dostanú na neúmernú úroveň.

Počítač je zložitá zostava rôznych jednotiek v hardware a software. Rôzne vzory a stupne vo vývoji vytvorili normy (ad hoc a zavedené), ktoré je možné nájsť všade v systéme. Vzhľadom k tejto nejednotnosti existuje veľký počet interfejsov.

Všetky tieto interfejsy a vzory rôznych kvalít a štýlov znamenajú problémy pre užívateľa alebo programátora pri použití

zariadenia - vyžaduje veľké vedomosti- a vzhľadom ku zložitosti môže programátor zaviesť skryté chyby.

Nedávno však boli vyvinuté takzvané redukované procesory. Redukovaný procesor vykonáva program, ktorý má určitú štruktúru, vrátane aritmetických výrazov a táto štruktúra je redukovaná v mnohých redukčných krokoch. Týmto nie je vykonávaný program v danej postupnosti ako u iných druhov počítačov.

Boli určité problémy pri vývoji redukovaného procesoru nad daný limit veľkosti.

Vývoj programovacích jazykov:

Vývoj prvého elektronického počítača začal vývoj niekoľkých jazykov, vhodný pre daný typ počítača ako napríklad FORTRAN, COBOL, Algol, BASIC, Pascal. Tieto jazyky sa nazývali imperatívne jazyky, ďalej tiež nazývané konvenčnými jazykmi hlavne preto, že normálne dávajú programy, ktoré sa skladajú z postupnosti povelov alebo inštrukcií, ktoré majú byť vykonávané sekvenčne konvenčným počítačom, to je počítačom, ktorý bol navrhnutý podľa princípov ako ich navrhol John von Neumanh. Narastajúca nepohodlnosť týchto jazykov viedla k vývoju ďalšej série jazykov: LISP, ISWIM, Scheme (dialekt LISPu), ML, Hope, SASL atď. Hnacou silou za týmto vývojom jazykov bola koncepčná jednoduchosť, žiadne konkrétne zariadenie neovplyvnilo návrh. Trvalo určitú dobu, kým sa funkčné jazyky dostali do pozornosti - jedným z dôvodov bolo, že funkčné programy mali dlhú dobu vykonávania. Neskorší vývoj dokázal, že rýchlosť vykonávania v niektorých prípadoch môže byť takmer alebo úplne rovnaká ako u konvenčných (imperatívnych) jazykov i keď tieto funkčné programy nie sú vytvorené pre to, aby boli vykonávané na tomto type počítača.

Kríza software:

Masívne úsilie o vývoj funkčných jazykov bolo podnietené zvyšujúcou sa nepohodlnosťou imperatívnych jazykov. O kríze software sa začalo hovoriť okolo roku 1970. Programy boli zložitejšie a často obsahovali mnoho chýb, ťažko sa čítali, ťažko sa chápali a zvlášť sa ťažko upravovali. Jeden z dôvodov bolo, že vysokoúrovňové imperatívne jazyky by zjednodušili programovanie boli nastavené príliš vysoko - tieto jazyky neboli na tak vysokej

úrovni alebo abstraktnej úrovni, ako sa možno zdalo. Imperatívne jazyky sú stále upravované podľa prvých koncepcií počítačov, von Neumannov typ počítača, a úroveň programovania je stále veľmi blízko strojovej úrovne. Funkčné programovacie jazyky majú niekoľko vlastností, ktoré zmierňujú niektoré nevýhody viac konvenčných jazykov.

Pre ďalšie informácie a pochopenie odkazujeme na príručku "Functional Programming Using Standard ML", Ake Wikstrom, Prentice Hall 1987.

Aby bolo možné celkom pochopiť ciele a výhody vynálezu, je nutné pochopiť čo obsahuje a funkčný prístup pri výpočte, zvlášť porovnanie s historicky obvyklejším imperatívnym prístupom.

Výraz "funkčný prístup" znamená, že programy sú napísané a uložené a vykonávané na počítači, ktorý zahrňuje hardware, vhodný zvlášť pre tento jazyk. Podobne výraz "imperatívny prístup" znamená, že programy sú napísané v imperatívnom jazyku a uložené a vykonávané na počítači, ktorý obsahuje hardware, vhodný pre imperatívne jazyky.

Je však možné uložiť a vykonať programy na konvenčnom počítači, napísané vo funkčnom jazyku. Možný je aj opak - programy, ktoré sú napísané v imperatívnom jazyku môžu byť urobené na počítači, ktorý je vhodný pre vykonávanie programov, napísaných vo funkčnom jazyku.

Vlastnosti funkčných jazykov:

Na program, ktorý je napísaný vo funkčnom jazyku, sa môže pozerať ako na súbor definícií vlastností objektov a ako na pravidlá výpočtov. Definícia je deklaratívna časť a pravidlá výpočtov alebo pravidlá pre redukciu alebo prepis sú operačná časť, ktorú počítač používa pri vykonávaní. Funkčné jazyky poskytujú interfejs vyššej úrovne pre počítač, ktorý umožňuje, aby sa programátor nezaoberal podrobnosťami, ktoré sa týkajú hardware počítača. Ako vedľajší efekt je, že funkčné programy sú často kratšie a ľahšie pochopiteľné ako konvenčné imperatívne programy. Jedna z hlavných nevýhod funkčných nevýhod

je to, že funkčné programy musia byť preložené do konvenčného jazyka, aby ich bolo možné vykonávať na konvenčnom počítači. To bolo vykonávané kompilátormi alebo interpretačnými programami. Je jasné, že niektoré z výhod funkčného prístupu boli potlačené vďaka skutočnosti, že neexistoval špeciálny hardware pre úlohu ukladania a vykonávania funkčných programov efektívnym spôsobom.

Podstata vynálezu

Hlavným účelom vynálezu je vytvoriť procesor, v ktorom by bola eliminovaná väčšia časť hardwarových obmedzení.

Ďalším cieľom vynálezu je vytvoriť veľmi rýchly procesor, v ktorom bude programovanie čo najjednoduchšie a zložitosť programovania bude tvorená jednotným a jednoduchým svetom.

Ešte ďalším cieľom vynálezu je vytvoriť procesor, ktorý bude súčasne prispôsobený používanému jazyku, ako strojový jazyk, operačný systém, komunikačný protokol a programovací jazyk, to je jazyk, ktorý je jednoduchý a v ktorom boli úrovne software privedené na minimum.

Ďalším cieľom software je vytvoriť procesor, ktorý je veľmi jednoduchý a flexibilný v použití ako pre programátorov tak aj pre systémových správcov návrhárov hardware a ľudí zaoberajúcimi sa súčiastkami.

Ďalším cieľom vynálezu je vytvoriť procesor, ktorý je prispôsobený jazyku deklaratívneho programovania, ktorý má chovanie ako sémantické objekty a tak popisuje vlastnosti reálneho času a systému.

A nakoniec ďalším cieľom vynálezu bolo vytvoriť procesor, ktorý má štrukturálnu aritmetickú jednotku, ktorá je oddelená od číslícovej aritmetickej jednotky a spolupracuje s ňou.

K dosiahnutiu vyššie uvedeného a ostatných cieľov berie vynález do úvahy redukčný procesor riadený programom, ktorý má štruktúru a je prispôsobený na redukciiu uvedenej štruktúry v niekoľkých redukčných krokoch, ktoré zahrňujú rôzne druhy redukcií, procesor prvého rádu tohoto typu zahrňuje aktívne ukladanie, zahrňujúce

- a. množinu aktívnych buniek na ukladanie, kde každá má možnosť zvýšiť výkon redukčnej operácie a
- b. komunikačnú sieť, po ktorej prebieha komunikácia s výsledkami každej redukcie smerom na všetky bunky medzi bunkami pre ukladanie, ktoré majú prepojenie s uvedenými výsledkami.

Komunikačná sieť obsahuje usporiadanie do zbernice, ktorá má riadiace a dátové vodiče, a všetky vodiče sú spojené s každou uvedenou pamäťovou bunkou. Každá pamäťová bunka môže byť schopná zahrnúť všetky informácie, ktoré sú nutné pre vykonanie redukčnej operácie.

Redukčná informácia môže tiež obsahovať referenciu minimálne na jednu z uvedených ostatných buniek, pričom obsah pamäťových buniek má väzbu na uvedenú referenciu v stromovej štruktúre. Minimálne jedna z uvedených pamäťových buniek, ktorá sa nazýva jadrová bunka, je schopná vykonať všetky druhy redukcií a zvyšok uvedených buniek, ktoré sa nazývajú objektové pamäťové bunky, je schopný vykonať iba obmedzené časti všetkých druhov redukcií. Objektové pamäťové bunky uloženia sú zahrnuté v asociatívnej pamäti, ktorá má usporiadanie prvej zbernice pre externé riadenie a usporiadanie druhej pamäťovej zbernice pre dáta, vrátane:

- niekoľkých uvedených objektových pamäťových buniek pre ukladanie kompozitných informácií,
- prostriedkov v každej z uvedených objektových pamäťových buniek pre ukladanie minimálne jednej značky, kde uvedené značky budú označovať minimálne zvolený stav(y) alebo nezvolený stav(y) pre uvedené objektové pamäťové bunky,
- prostriedkov pre vykonanie operácií vyhľadávania medzi uvedenými objektovými pamäťovými bunkami k nastaveniu

uvedených značiek a dekodér priority, s ktorým sú všetky uvedené objektové pamäťové bunky zviazané a ktoré zvolia jednu z niekoľkých uvedených objektových pamäťových buniek.

Existuje minimálne jedna globálna zbernica pre vykonanie logických operácií typu AND a OR medzi uvedenými pamäťovými bunkami a prostriedky v každej pamäťovej bunke pre komunikáciu s uvedenými zbernicami a pre riadenie uvedených pamäťových buniek, ktoré sa odohrávajú pri konkrétnej logickej operácii.

Niekoľko redukčných procesorov prvého rádu môže byť spojených do štvorcového poľa tak, že predstavujú redukčný procesor druhého rádu, kde každý procesor prvého rádu v uvedenom štvorcovom poli zahrňuje kanál, ktorý ide ku každému susednému redukčnému procesoru prvého rádu v uvedenom štvorcovom poli.

Redukčný procesor druhého rádu naopak môže byť rozdelený na logické oblasti, ktoré majú rozmer 2^n krát 2^n redukčných procesorov prvého rádu, uvedené logické oblasti majú na vzájomných stranách pravidelný vzorec, aby sa pokrylo uvedené pole uvedených redukčných procesorov druhého rádu.

Minimálne jeden pripísateľný prvok, ktorý obsahuje adresu, je uložený v každej oblasti, uvedený prvok je pripísateľný z pamäťovej bunky v oblasti, do ktorej patrí.

Procesor prvého rádu môže byť pripojený k inému takému procesoru druhého rádu tak, že každú uvedenú oblasť v rámci uvedeného procesoru druhého rádu je možné premiestniť o polovicu oblasti v každom smere vnútorným predefinovaním rozmiestnenia v uvedenej oblasti v rámci uvedeného procesoru druhého rádu.

Alternatívne je možné pripojiť niekoľko uvedených redukčných procesorov prvého rádu v hierarchii siete, pričom každá sieť je zbernica alebo okruh, alebo niekoľko uvedených procesorov prvého rádu môže byť zapojených v hierarchii sietí, pričom sú tu minimálne dva druhy sietí, prvý druh bude zbernica, druhý druh bude okruh a tretí druh bude štvorcové pole.

Minimálne jeden prostriedok portu je pripojený k uvedeným prostriedkom aktívneho ukladania a minimálne jeden z obklopujúcich prostriedkov je pripojený k minimálne jednému

z uvedených prostriedkov portov.

Prostriedky sú zaistené pre porovnávanie postupnosti signálu na prostriedkoch portov s postupnosťou, ktorá je uložená v minimálne jednej pamäťovej bunke, pričom uložená postupnosť umožňuje nedefinovanú postupnosť elementov v uvedených pamäťových prvkoch, to znamená niečo, čo predstavuje rozpor, pokiaľ porovnanie vedie k jasnej rozdielnosti, alebo inak prepísať uvedenú uloženú postupnosť do určenej postupnosti, ktorá je zjednotením signálu postupnosti od okolia a uloženej postupnosti z počítača.

Množina redukčných procesorov druhého rádu, geograficky vzájomne od seba oddelených, môže tvoriť redukčný procesor tretieho rádu.

Asociatívne adresovanie, obmedzené podľa oblasti, je prevzaté pre redukčné procesory druhého rádu a fyzické adresovanie je prevzaté pre procesory tretieho rádu.

Vykonávanie redukcií:

Program, ktorý má byť urobený, môže byť znázornený smerovým grafom zopnutia, kde každá časť programu je znázornená zopnutím. V priebehu vykonávania tento smerový graf zopnutia sa postupne redukuje podľa pravidiel redukovania použitého jazyka. Kde nezostali žiadne vykonávateľné zopnutia, vykonávanie programu je skončené. Na smerový graf zopnutia je možné pozeráť ako na stromovú štruktúru, kde každý uzol stromu je zopnutie a kde uzol, ktorý je najvyššie, sa nazýva koreň. Vykonávanie redukcie je potom normálne vykonávané redukciou stromovej štruktúry hore nohami, pričom sa redukujú časti stromu, ktoré sú ďalej od koreňa a potom sa pokračuje ďalej smerom ku koreňu. Tento druh vykonávania sa bežne volá vykonávanie riadené povelom, to znamená vykonávanie častí programu v závislosti na výsledku vykonávania ostatných častí je odložené do doby, kým je k dispozícii výsledok.

Definície:

V nasledujúcom je uvedený zoznam výrazov, používaných v tejto špecifikácii a ich význam:

Prvok - časť niečoho väčšieho v štruktúre dát.

Zoznam - usporiadaná postupnosť prvkov, kde každý prvok môže naopak byť zoznamom.

Vložený zoznam - časť zoznamu dostatočne malý k tomu, aby bol uložený ako celok v jednom bunkovom zopnutí. Umožňuje znázornenie ľubovoľne dlhého zoznamu.

Zopnutie - hierarchicky štrukturovaná entita, ktorá definuje proces. Všetky zopnutia majú koreň, ktorý jednoznačne definuje zopnutie. Redukčné práce v redukčnom počítači sa vykonávajú na zopnutiach. Celý stav počítača sa transformuje redukciami.

Objektové ukladanie - pamäť vrátane pamäťových buniek, ktoré si pamätajú objekt.

Pamäťová bunka - bunka v objektovom ukladaní. Pamätá si zopnutie bunky, ktorá môže mať vzťah k ostatným zopnutiam buniek, uložených v ostatných pamäťových bunkách.

Zopnutie bunky - obsah pamätevej bunky.

Pole pamäťových buniek - pole v pamätevej bunke.

Prvok zopnutia - dátový element uložený v poli buniek.

Identifikátor zopnutia - prvok bunky zopnutia jednoznačne popisujúci zopnutie.

Kanonické zopnutie - zopnutie, ktoré nemôže byť ďalej redukované, teda zopnutie bunky, ktoré neobsahuje žiadne identifikátory zopnutia určujúce niektoré iné zopnutia bunky, ktoré by mohlo byť redukované takým spôsobom, že toto zopnutie bunky by malo byť ďalej redukované.

Objekt - zopnutie, ktoré má byť vykonané, teda redukované.

Otec - zopnutie, ktoré má minimálne jeden identifikátor zopnutia v poli hodnota/popis.

Syn - zopnutie zviazané s iným zopnutím cez identifikátor zopnutia, ktoré určuje syna. Syn môže byť tiež otcom. Otec môže byť tiež synom. Syn môže mať viac ako jedného syna.

Sieť zopnutia - zopnutie a všetci jeho otcovia.

Poloha zopnutia - ak zopnutie je koreň alebo uzol.

Koreň - najvyššie zopnutie v strome zopnutia.

Uzol - zopnutie v strome zopnutia, ktoré nie je koreň.

Kde - pole pamäťových buniek, ktoré obsahujú polohu zopnutia.

Typ - kód typu v bunke zopnutia, to je bitový vzor, ktorý predstavuje vlastnosť objektu, to je inštrukčný kód.

Zdlhavý - prvok v bunke zopnutia, ktorý udáva, či je vykonávateľný alebo s odloženým vykonávaním alebo neaktívny.

Identifikátor - zvláštny druh spínacieho prvku, ktorý sa používa na označenie objektu, uloženého v pamäťovej bunke.

Prostredie - objekty môžu byť zoskupované tak, že sa im priradí rovnaké prostredie.

Hodnota/popis - prvok zopnutia, v ktorom je uložená buď jednoduchá hodnota, to je priama reprezentácia nič alebo popis iného zopnutia, to je nepriama reprezentácia.

Jadrová bunka - štrukturálna aritmetická jednotka. Jadrová bunka je schopná vykonávať štrukturálnu aritmetiku, ktorá znamená redukčné zopnutie.

Číslicová ALU - číslicová aritmetická jednotka, schopná vykonávať základné aritmetické a logické operácie. Jadrová bunka využíva číslicové ALU pre číselné operácie.

Plný register - register, ktorý sa rozprestiera cez celé roviny v jadrových bunkách.

Jadrové slovo - obsah plného registra v jadrovej bunke.

Obmedzený register - register v jadrovej bunke, ktorý sa rozprestiera v obmedzenom počte rovín, ktoré majú taký rozmer, aby zahrnuli prvok bunky zopnutia pre typ hodnoty/popisu.

Prvkové slovo - obsah obmedzeného registra alebo jeho časti, ktorý má rovnaké rozšírenie ako obmedzený register.

Číslicové slovo - časť prvkového slova, ktoré predstavuje hodnotu alebo popis.

Príznakové slovo - časť prvkového slova, ktoré má príznak, ktorý označuje vlastnosť reprezentácie číslicového slova.

Redukcia - prepísanie/reštrukturalizácia zopnutia v súlade s pravidlami pre použitie konkrétneho jazyka pre programovanie.

Asociatívne objektové ukladanie:

Stratégia pri odvodzovaní procesoru podľa vynálezu prvého

rádu bola vytvoriť zariadenie pre objektové ukladanie ako niekoľko pamäťových buniek. Každá taká bunka by mala obsahovať jedno bunkové zopnutie alebo by mala byť voľná nepoužitá bunka. Bunky nesmú byť priradené v žiadnom konkrétnom poradí, ale namiesto toho sa predpokladá, že sú nádržou dostupných zdrojov.

Predpokladalo sa, že je veľmi dôležité nemať žiadnu fyzickú adresu alebo závislosť na fyzickom umiestnení. Také závislosti by skôr alebo neskôršie spôsobili problémy vo všetkých konvenčných typoch pamäte RAM.

Všetky bunky v objektovom ukladaní komunikujú cez pamäťovú zbernicu. To je dôležité preto, aby sa znížili náklady. Ostatné usporiadania, ako použitie niekoľkých portov atď., zvýši oblasť objektového ukladania. Usporiadanie s pamäťovou zbernicou však môže vykonať iba jednu operáciu v priebehu pamäťového cyklu.

Redukčný mechanizmus bude mať stav, ktorý sa skladá z bunkových zopnutí, z ktorých každý má identifikátor a kde bunkové zopnutia, zviazané spolu s identifikátorom, tvoria graf. Graf môže byť riadený adresovými bunkovými zopnutiami podľa ich identifikátorov. Preto pamäťová zbernica bude používaná ako cesta zdieľania času pre všetky vrcholy grafu.

Každé bunkové zopnutie má tiež prostredie, ktoré môže obsahovať identifikátor, ktorý označuje koreňové zopnutie v strome a pri tom udáva prostredie pre zopnutie. Týmto spôsobom je celá štruktúra prístupná z jedného zopnutia v tomto strome cez koreň iba jednou operáciou. Zopnutia môžu byť zoskupené spoločne podľa rovnakého prostredia.

Každé "adresovanie" musí byť vykonané podľa obsahovej informácie, pretože tu nie sú žiadne fyzické závislosti, to je že pamäť je asociatívna. Bunkové zopnutie, teda obsah v pamäťovej bunke, zahŕňa niekoľko prvkov zopnutia, ktoré sú uložitelné v pamäťových poliach v pamäťovej bunke. Každý pamäťový prvok môže obsahovať identifikátor spoločne s informáciou návestia alebo reprezentáciu hodnoty alebo indikáciu toho, že nie je používaný. Pri asociatívnom vyhľadávaní je možné použiť ľubovoľný

prvok zopnutia alebo kombináciu prvkov zopnutia ako je vyhľadávací kľúč to je identifikátor bunky, prostredie bunky, typ bunky, hodnoty zapísané do polí pamäťovej bunky alebo ich kombinácia.

Prvky zopnutia môžu obsahovať zvláštny výberový bit, ktorý udáva, že prvok zopnutia bol zvolený ako cieľ pre mechanizmus prístupu. Určité operácie vyhľadávania nastavujú tieto zvolené bity v poliach pamäťovej bunky.

Tento typ prístupu môže vyžadovať jednu alebo viac pamäťových buniek. Jedna operácia s mnohými bunkami môže byť taká, že ukladá identifikátor v mnohých zvolených pamäťových poliach, ktoré patria k rôznym pamäťovým bunkám.

Centrálna riadiaca jednotka.

Základných pravidiel pre redukciu je toľko, že nie je možné ich zahrnúť do každej pamäťovej bunky. Preto je redukčný mechanizmus zdieľaný všetkými bunkovými zopnutiami. Používajú sa riadiace a snímacie zosilovače, aby všetky objektové pamäťové bunky boli schopné riadiť dlhé spoje pamäťovej zbernice. Centrálna riadiaca jednotka môže nastaviť signály ako v čase tak i v úrovni. Preto všetky komunikácie sú organizované cez usporiadanie pamäťovej zbernice, pričom centrálna riadiaca jednotka ovláda zariadenie. Centrálna riadiaca jednotka, ktorú by malo predstavovať boolovské hradlové pole, umožňuje celkom jednoduché dekódovanie hodnoty na externých vodičoch a vysiela väčšinu riadiacich signálov na všetky pamäťové bunky v objektovom ukladaní cez usporiadanie riadiacej zbernice. Upozorňujeme na príručku "Introduction to VLSI Systems", ktorej autori sú Carver Mead a Lynn Conway z nakladateľstva Addison Wesley Publishing Company, 1980, kde nájdete ďalšie informácie, týkajúce sa návrhu riadiacich jednotiek.

Aby bolo možné vykonávať zložité vzory prístupu, boolovské výrazy sú riešené v každej pamäťovej bunke. Tento výraz je riadený vysielaným riadiacim slovom z centrálnej riadiacej jednotky. Riadiace slovo je normálne odvodené centrálnou riadiacou

jednotkou zobratím informácie v pamäťovej bunke. Pre väčšinu aplikácií redukčného procesoru je dôležité, aby boolovské výrazy boli riešené v pamäťových bunkách, pretože to bude omnoho efektívnejšie čítať informácie z pamäte bežným spôsobom a potom ich spracovať.

Boolovské výrazy sú zvolené tak, aby boli, pokiaľ je to možné, čo najmenšie, aby nimi spôsobený priestor bol čo najmenší. Preto sú teda riadiace funkcie pozorne vyberané.

Takto riadi riadiaca jednotka funkciu objektového ukladania. Usporiadanie pamäťovej zbernice komunikuje so všetkými pamäťovými bunkami. V niektorých prípadoch bude preto niekoľko pamäťových buniek určených na čítanie. K tomu účelu je tu mechanizmus pre výber iba jednej z nich z niekoľko možných v danom čase. Výber je vykonaný prioritným dekodérom, ktorý je pripojený ku všetkým pamäťovým bunkám.

Jadrová bunka.

Konkrétna aktívna pamäťová bunka, ktorá sa ďalej nazýva jadrová bunka, najlepšie spolupracuje s objektovým ukladáním, ktoré je potom považované za aktívne ukladanie. Jadrová bunka je schopná vykonávať všetky typy redukcií, zatiaľ čo asociatívne objektové ukladanie obsahuje pamäťové bunky, schopné vykonávať iba obmedzené časti niektorých typov redukcií. Ale usporiadanie jadrovej bunky môže byť simulované v časti objektového ukladania - bude však menej účinné.

Jadrová bunka je schopná uložiť množinové úrovne výrazu. Základná inštrukcia vo výraze, uloženom v jadrovej bunke, musí byť vykonaná v jadrovej bunke, to je aritmetika pre prepísanie štruktúry, ktorá je daná programom počítača. Výraz bunky v jednotke má veľkosť, ktorá sa týka operácie prepísania.

Číslicovo/štruktúrálna aritmetická jednotka.

Je nutné poznamenať, že jadrová bunka je zariadenie, ktoré iba vykonáva štruktúrálnu aritmetiku, to je väčšinu redukcií a že využíva číslicovú aritmetickú jednotku (číslícová ALU) pre

vykonanie číslícových operácií. Normálne je procesor vybavený aritmeticko-logickou jednotkou (ALU), ktorá vykonáva ako číslícové tak i časti štruktúrálnych operácií a táto ALU nie je žiadnym spôsobom rozdelená na rôzne funkčné časti pre štruktúrne a číselné operácie. Keď sa používa konvenčné ALU, je potrebný malý program, aby sa vykonali operácie ekvivalentné ku štruktúre operácií vykonávaných v jadrovej bunke.

Prehľad obrázkov na výkrese

Pre lepšie komplexné pochopenie predkladaného vynálezu a pre ďalšie účely a výhody sa vykonávajú referencie na nasledujúce popisy v spojení s priloženými výkresmi, kde obr.1 predstavuje schématicky štruktúru redukčného procesoru prvého rádu podľa vynálezu, obr.2 schématicky znázorňuje návrh použitia pamäte do redukčného procesoru prvého rádu na obr.1, obr.3 schématicky znázorňuje použitie pamäťovej bunky do pamäte podľa obr.2, obr.4A ukazuje použitie rôznych polí v pamäťovej bunke, ktorá môže mať uložené bunkové zopnutie, obr.4B až 4D ukazujú jadrové registre, ktoré môžu byť použité pri použití jadrovej bunky, obr.4E ukazuje možnú konfiguráciu jadrových registrov v použití jadrovej bunky, obr.5A až 5F ukazuje rôzne formy uloženia dát v jadrovej bunke, obr.6 až 8 ukazuje príklady operácie jadrovej bunky, obr.9 ukazuje blokovú schému použitia operačnej roviny v jadrovej bunke v redukčnom procesore prvého rádu z obrázku 1, obr.10 ukazuje schématicky použitie návrhu generickej registrovej bunky v jadrovej bunke, ktorá má všetky možné prepojenia pre operačnú rovinu podľa obrázku 9, obr.11 a 12 ukazuje schématicky dva príklady prenosu medzi registrovými bunkami v jadrovej bunke, obr.13 až 19 ukazuje rôzne príklady prenosu dát vo vnútri jadrovej bunky, obr. 20 ukazuje obvodovú schému bitovej bunky v pamäťovej bunke (ukázané na obr. 3) a riadiace a snímacie obvody, ktoré sú k nej pripojené, obr.21A ukazuje obvodovú schému bloku, ktorý je súčasťou dekodéru priority

v pamäti, obr.21B znázorňuje prepojenie blokov, ukázaných na obr.21A, obr. 22 ukazuje obvodovú schému použitia prvkovej hlavy v pamäti, obr.23 ukazuje obvodovú schému použitia spínacej hlavy v pamäti, obr.24A ukazuje obvodovú schému číslicovej ALU v procesore prvého rádu z obr. 1, obr. 24B ukazuje schému hodinového signálu a niektoré základné riadiace signály pre číslicovú ALU podľa obr.24A, obr.25 ukazuje schématickú reprezentáciu rozdelenia bitového poľa v aritmetickej reprezentácii, obr.26A ukazuje grafickú reprezentáciu pre celé čísla, obr.26B ukazuje grafickú reprezentáciu kódu pre hodnoty s plávajúcou rádovou čiarkou, obr.26C ukazuje schématickú reprezentáciu pre vysvetlenie nastavenia aritmetickej reprezentácie, vykonanej na 7-bitovej zbernici, obr.27A až D ukazuje zoznam dát, platný pre obrázok 26, obr.28 ukazuje obvodovú schému použitia číslicovej ALU v procesore prvého rádu podľa vynálezu, obr.29A až C predstavuje použitie vstupného/výstupného bufferu číslicovej ALU v rozdielnych hodinových sekvenciách, obr.30 predstavuje obvodovú schému použitia presného dekodéru obsiahnutého v obvode z obr.28, obr.31A predstavuje blokovú schému použitia sčítačky plného slova do obvodu podľa obr.28, obr.31B predstavuje blokovú schému presného rezu v sčítačke podľa obr.31A, obr.31C predstavuje blokovú schému bitového rezu v presnom reze podľa obr.31B, obr.32 predstavuje použitie presného rezu v inkrementačnom člene obvodu podľa obr.28, obr.33 predstavuje obvodovú schému použitia základného bitového rezu voliča, ktorý je zahrnutý vo voliči v operačných obvodoch v obvode podľa obr.28, obr.34 je schéma signálovej postupnosti podľa vynálezu, obr.35 ukazuje použitie portu do a/alebo výstup z procesoru podľa vynálezu, obr.36 predstavuje veľmi všeobecne redukčný procesor druhého rádu (SOP) podľa vynálezu, obr.37A ukazuje na použitie procesoru SOP, kde sú procesory prvého rádu (FOP) usporiadané v štvorcovom poli, obr.37B predstavuje operačnú rovinu v jadrovej bunke opatrenú štyrmi samostatnými prostriedkami pre prenos dát pre možné použitie v štvorcovom poli procesoru SOP z obr.37A,

obr.37C predstavuje štvorcové pole procesoru SOP vrátane 256 procesorov FOP rozdelených na štvorcové oblasti, obr.37D až 37E znázorňuje rôzne mäkké okruhy v štvorcovom poli procesora SOP, obr.38 predstavuje druhé použitie procesore SOP, kde sú procesory FOP usporiadané v hierarchii siete, kde každá sieť je zbernica, obr.39 predstavuje tretie použitie procesoru SOP, kde procesory FOP sú usporiadané v hierarchii siete, kde sieť je tvrdý okruh, obr.40 ukazuje použitie kombinácie procesorov SOP, zahrňujúci SOP z obr.37A, 38 a 39, obr.41 predstavuje redukčný procesor tretieho rádu, ktorý obsahuje niekoľko SOP, obr.42A až C predstavuje rôzne časti redukčného procesoru tretieho rádu a znázorňuje jeden spôsob ako ovládať globálne adresovanie.

Príklady uskutočniteľnosti vynálezu

Redukčný procesor prvého rádu.

Na obr.1 je ukázané použitie redukčného procesoru podľa vynálezu. Procesor prvého rádu je redukčný procesor, ktorý môže byť použitý ako časť v zložitejšom redukčnom procesore, ako bude popísané ďalej. Schématická štruktúra tohoto redukčného procesoru je zrejmá z obr.1. Je však nutné poznamenať, že jeho návrh by mohol byť úplne iný, napríklad rôzne roviny jadrovej bunky by mohli byť vykonané v paralelných vrstvách, ale nebudú mať presne rovnaký návrh ako je schématicky znázornené.

Redukčný procesor prvého rádu na obr.1 zahrňuje aktívnu asociatívnu pamäť 1, ďalej nazývanú objektová pamäť, ktorá zahrňuje pamäťové bunky ďalej nazývané pamäťové bunky, z ktorých každá je schopná uložiť dátový objekt, ktorý môže zahrňovať vykonávateľné časti a tiež je prednostne schopná poskytnúť obmedzené súbory redukčnej aritmetiky, štrukturálnu aritmetickú jednotku 2 ďalej nazývanú jadrová bunka, vrátane niekoľkých jadrových registrov 3 na vykonávanie štrukturálnych aritmetických operácií, to je redukcie, ktoré modifikujú štruktúru a pokiaľ je redukčný procesor

prvého rádu z obr.1 súčasťou redukčného procesoru druhého alebo viacnásobného rádu vrátane niekoľkých redukčných procesorov prvého rádu, prostriedky 4 pre prenos dát sieťami procesoru, ktoré vykonávajú komunikáciu medzi viacnásobnými procesormi. Prostriedky 4 pre prenos dát zahrňujú niekoľko registrov, prispôsobených na dočasné uloženie štruktúry pamätevej bunky, aby sa táto štruktúra mohla preniesť do iného procesoru prvého rádu.

Je možné rezervovať pole v objektovej pamäti pre operácie, ktoré sú vykonávané v jadrových bunkách, to je že jadrové bunky môžu byť simulované v časti objektovej pamäti. Jadrová bunka má niekoľko funkcií pre výmenu a prenos dát medzi svojimi registrami iba v jednej operácii. Také funkcie zaberú niekoľko operačných cyklov v poli objektového ukladania, ktoré simuluje jadrovú bunku.

Číslicová aritmeticko-logická jednotka (číslicová ALU) 5 je pripojená k jadrovej bunke 2 a dáta z jadrovej bunky sú prenášané do číslicovej ALU keď sa majú vykonávať číselné operácie. Takto sú štrukturálna a číslicová aritmetika rozlíšené, samostatné jednotky.

Procesor tiež zahŕňa centrálnu riadiacu jednotku 6 pre riadenie prvkov v objektovej pamäti 1, jadrovej bunke 2 a číslicové ALU 5. Ďalej môže byť procesor vybavený niekoľkými portami pre komunikáciu s vonkajším okolím.

Centrálna riadiaca jednotka.

Centrálnou riadiacou jednotka 6 je s výhodou boolovské hradlové pole pomerne zložitej konfigurácie. Jeho použitie nie je možné ukázať, pretože jeho konfigurácia je vypočítavaná na počítači, ktorý je vybavený všetkými algoritmami, ktoré zohľadňujú konfiguráciu riadiacich signálov vo vzťahu k informatívnym signálom. Boolovské hradlové pole je uskutočnené priamo ako maskové vzory čipu z počítača, ktorý je prispôsobený konfigurácii maskovým vzorom čipu pre celý procesor podľa vynálezu. Preto tento

obvod nemôže byť ukázaný podrobne. Iba signály ktoré smerujú z nej a do nej môžu byť popísané v spojitosti s použitiami popísanými ďalej.

Objektová pamäť.

Objektová pamäť 1 má podstatne viac inteligencie ako bežná pamäť typu RAM. Je asociatívna, čo umožňuje, aby poskytovala viac služieb ako iba "čítanie" a "zápis", čo umožňuje bežná pamäť typu RAM - bude vysvetlené ďalej. Štruktúra bitovej bunky, vhodná zvlášť pre objektové ukladanie, je ukázaná na obr. 20.

Pamäťová bitová bunka je schopná vykonávať mnoho funkcií, aj keď obsahuje iba štyri prípoje, z ktorých tri sú ovládateľné. Obsahuje veľmi málo súčiastok. To dáva možnosť vytvoriť kompaktné zariadenie pre ukladanie vrátane obrovského množstva pamäťových bitových buniek.

Objektové ukladanie je rozdelené v pamäťových bunkách, z ktorých každá obsahuje spínacie prvky. Poskytované služby sú na vysokej úrovni. Napríklad je možné nájsť všetky výskyty konkrétneho dátového prvku nech má akékoľvek umiestnenie v rámci jednotlivcej pamäťovej bunky a globálne prepísať konkrétny dátový prvok, to je v rámci celého objektového ukladania na novú hodnotu s použitím iba jednej pamäťovej inštrukcie. Pretože objektové ukladanie je asociatívne, táto operácia prepísania môže byť vykonaná v dvoch fyzických pamäťových cykloch nezávisle na počte dotknutých pamäťových buniek.

Podľa obrázku 2 asociatívne objektové ukladanie 1 sa skladá z pamäťovej roviny rad pamäťových buniek 10. Tak sú pamäťové bunky vytvorené ako rady v zásobníkovej pamäti. Všetky sú pripojené k vertikálnej pamäťovej zbernici t_1 , t_2 , id, env, v_0 , v_1 , v_2 , v_3 riadenej riadiacimi a snímacími zosilovačmi v transformačnom interfejsu 7, ako bude vysvetlené neskoršie vo vzťahu k obr. 20. Všetky pamäťové bunky sú tiež pripojené k dekodéru priority 11, ktorý zvolí jednu pamäťovú bunku 10 v danom okamžiku z niekoľkých pamäťových buniek, ktoré majú byť ovládané. Objektové

ukladanie je riadené asociatívnym prístupom, vykonávaným počas relácie niekoľkých málo cyklov. Prístup s plným objektovým ukladáním sa vykonáva počas relácie.

Port.

Aby sa mohlo komunikovať s vonkajším okolím, redukčný procesor musí byť vybavený jedným alebo niekoľkými portami pre vstup/výstup dát. Port je možné vidieť ako predĺženie objektového ukladania, pretože obsahuje pamäťové bunky, ktoré sú funkčne podobné a kompaktilné s bunkami pre objektové ukladanie. Port vykonáva napriek zjednoteniu vstupno/výstupné údaje.

Objektové pamäťové bunky.

Pamäťová bunka 10 v objektovom ukladaní 1 je použitá ako k uloženiu digitálneho obsahu, tak i k tomu, aby prevzala aktívnu účasť pri konkrétnom výpočte. Zložená digitálna informácia a minimálne jedna značka sú uložené v každej pamäťovej bunke 10. Značky sú buď CHOSEN (vybrané) alebo NON CHOSEN (nevybrané).

Zložená digitálna informácia v pamäťových bunkách alebo v ich časti, nazývané spínacie prvky, môže byť čítaná alebo zapísaná čítaním alebo zápisom informácie, ktorá je označená ako CHOSEN. Prístup môže tiež byť zaistený bez toho, aby sa použil bit značky. Prístup je potom riadený výsledkom logickej funkcie na jednobitovej zbernici a a b, pripojenej k prvkom, ako je ukázané na obr. 3, ktorý predstavuje pamäťové bunky.

Dve globálne jednobitové zbernice 12 (MODE) a 13 (MORE) sú k dispozícii pre vykonanie logických operácií medzi pamäťovými bunkami 1. Tieto operácie môžu byť AND a OR. Tretia zbernica 14 (ANY) môže byť pripojená k dekodéru priority. Avšak počet zberníc nie je obmedzený na tri, ale môže byť iba jedna alebo niekoľko. Samostatné pamäťové bunky môžu čítať zo zbernice 12, 13 a 14 a tiež sa zúčastniť na logických operáciách. Zbernica 12 (MODE) bude mať signál v úrovni "áno" pokiaľ sa má vykonať test na ľubovoľnej zbernici a alebo b. Na zbernici 12 (MODE) je možné

vykonávať čítanie alebo zápis všetkými pamäťovými bunkami. Zbernica 13 (MORE) bude v stave hodnoty "áno" pokiaľ je vybraná jedna alebo MORE (viac) pamäťových buniek. Zbernica 13 (ANY) bude mať signál v stave "áno" (to je "1"), pokiaľ ANY (ľubovoľná) pamäťová bunka žiada komunikáciu.

Pamäťová bunka znázornená na obr.3 je rozdelená na niekoľko polí pamätevej bunky 151. Každé pole pamätevej bunky obsahuje niekoľko bitových jednotiek 15 a prvkovú hlavu 16. Spínací prvok je uložený v bitovej bunke 15 spoločne so značkou, ktorá predstavuje CHOSEN alebo NOT CHOSEN, uložených v prvkovej hlave 16. Spínacie prvky môžu mať priradené vzájomne odlišné úlohy. Dĺžka slova, ktorá má byť uložená v každom spínacom prvku, môže byť napríklad v ráde 38 bitov. Okrem značky v hlave 16 môžu byť použité tiež niektoré bity v bitových bunkách 15 ako informácie, týkajúce sa použitia zvyšku informácie v prvku. Preto takzvané označené slová môžu byť umiestnené v týchto bitoch. Šesť bitov môže byť použitých ako označené slová a 32 bitov ako normálna uložená informácia v bitových bunkách 15.

Ako je zrejmé z obr.3, všetky spínacie prvky nemusia mať rovnaký rozmer. Preto spínacie prvky, ktoré sú pripojené k častiam zbernice t1 a t2, to je TYP, sú menšie ako spínací prvok, ktorý je pripojený ku zvyšku častí zbernice. Prvková hlava 16 polí pamätevej bunky 151 je pripojená k lokálnym jednobitovým zberniciam a a b pamäťových buniek. Počet týchto zberníc môže byť zvolený ľubovoľným spôsobom. Vhodné je, aby bola minimálne jedna zbernica. Logické operácie typu WIRED AND a WIRED OR medzi CHOSEN spínacích prvkov využívajú tieto zbernice.

Každá pamäťová bunka má spínaciu hlavu 17, ku ktorej sú pripojené zbernice a a b. Spínacia hlava 17 je tiež pripojená ku dekodéru priority s minimálne jednou zbernicou o šírke jeden bit, čo v uvedenom príklade sú dve zbernice 18 a 19 a je tiež pripojený ku globálnym zberniciam 12 a 13. Spínacia hlava 17 slúži ako buffer. Bunky môžu čítať výsledok na týchto zberniciach alebo sa zúčastniť logickej operácie.

Použitie objektových pamäťových buniek.

NA obr. 4A je graficky ukázané použitie pamäťovej bunky a bude použité k vysvetleniu funkcie rôznych pamäťových polí v pamäťovej bunke, ako sú použité v procesore podľa vynálezu. Polia v pamäťovej bunke na obr. 4A nemajú rovnaký rád ako polia v pamäťovej bunke na obr. 3, pretože obr. 3 ukazuje hardware a obr. 4A použitie bunky. Ako je znázornené na obr. 4A, pamäťová bunka môže mať uložené dva druhy spínacích prvkov a obsahuje pamäťové polia konkrétne prispôbosené prvkom, ktoré majú byť uložené. Týmto poliam boli na obr. 4A dané rovnaké mená ako prvky, ktoré v ňom majú byť uložené.

Prvý druh spínacích prvkov.

Prvý druh prvkov popisuje rôzne stavy pamäťovej bunky. Jeden prvok tohoto druhu je LAZY (zdĺhavý), ktorý udáva, či je bunka v kľudovom stave, kedy je na zvyšok informácie v bunke pozerané ako na pasívnu informáciu, alebo v stave exec, to je vo vykonávateľnom stave alebo v stave wait (čakanie), to je vyhodnotenie bunky bolo odložené a čaká stále na výsledok, kedy bude môcť byť vykonaný. Pole LAZY obsahuje dva bity. Iný prvý druh prvku je WHERE (kde), ktorý udáva, či je spínač v koreni alebo v uzle stromu. Obsahuje jeden bit. Iný prvý druh prvku je TYPE, ktorý obsahuje kód (par, seq, apply, list, unify, atď). V uvedenom príklade obsahuje tri bity. Prvý druh prvkov je prispôbosený k uloženiu častí jadra registrov 3 v rovinách LAZY, WHERE a TYPE, uvedených na obr. 1, kedy je spínanie prenesené na jadrovú bunku z dôvodu redukcie. Preto boli rovnako popísané.

Jadrová bunka má však extra rovinu, ktorá sa nazýva CLOS/SIMPLE a v ktorej je udané, či informácia v registri je spínanie alebo normálna hodnota. Rovina CLOS/SIMPLE, LAZY, WHERE a TYPE sa ďalej nazývajú roviny ATTRIBUTE. V závislosti na aplikácii je možné mať viac rovín jadrových buniek.

Druhý druh spínacích prvkov.

Druhý druh spínacích prvkov popisuje identifikáciu, prostredie alebo hodnotu. Sú to IDENTIFIER, ENVIRONMENT, VALUE/DES. Tento druhý druh prvkov je prispôsobený k tomu, aby bol uložený v častiach jadrových registrov 3, ktoré sú umiestnené v rovinách HEAD a NUM, uvedených na obr.1. Každý z týchto prvkov obsahuje prvok slova, ktoré v uvedenom príklade obsahuje 38 bitov. Prvkové slovo naopak je rozdelené na číslícové slovo, ktoré má výhodne 32 bitov pre uloženie v rovinách NUM v jadrových bunkách a príznakové slovo, ktoré má s výhodou 6 bitov, ktoré majú byť uložené v rovinách HEAD, keď sú prenášané do jadrovej bunky. V závislosti na aplikácii je možné vytvoriť ďalšie roviny.

Príznakové slovo.

Každý spínací prvok druhého rádu má príznakové slovo, ktoré udáva charakter číslícového slova. Sú dva druhy týchto príznakov a to nepriame príznaky, teda príznaky, ktoré sa používajú pre identifikátory a prostredie a priame príznaky, teda príznaky, ktoré sa používajú pre jednoduché hodnoty alebo podobne. Príznaky nepriamych príznakov sú cls, canon a open. Ak príznak je cls znamená to, že číslícové slovo predstavuje spínač, ktorý môže byť redukovateľný. Pokiaľ príznak je canon, znamená to, že číslícové slovo predstavuje spínač, ktorý nemôže byť ďalej redukovaný. Pokiaľ príznakové slovo je open, znamená to, že pole identifikátora predstavuje spínač, v ktorom je vložený zoznam. Príklady priamych príznakov sú discr, cont, unused a nothing. Keď je príznakové slovo discr, znamená to, že číslícové slovo je typu integer. Keď je príznakové slovo cont, znamená to, že číslícové slovo je hodnota s plávajúcou rádovou čiarkou. Ak je príznakové slovo unused, znamená to, že číslícové slovo v poli postráda význam. Pokiaľ príznakové slovo je nothing, znamená to, že číslícové slovo v poli nepredstavuje nič, to je rozpor, napríklad zjednotenie spínača vrátane poľa, označeného ako nothing (nič), bude vždy nič. V závislosti na aplikácii je možné vytvoriť

ďalšie nepriame alebo priame príznakové slová.

Identifikátory.

Pokiaľ identifikátor poľa v pamäťovej bunke obsahuje prvok identifikátora, bunkový spínač v tejto pamäťovej bunke môže byť prenesený do jadrovej bunky. Každé z týchto polí pamäťových buniek VALUE/DES môže obsahovať identifikátor, označujúci iný bunkový spínač a tak vytvára spojenie s týmto ďalším bunkovým spínačom. Počet uložených spínačov môže byť znázornený ako smerový graf alebo strom bunkových spínačov, ktoré sú spojené dohromady pomocou identifikátora.

Prostredie.

Polia prostredia môžu obsahovať identifikátor, ktorý označuje koreňový spínač v strome spínačov, ktoré vytvára prostredie spínača. Týmto spôsobom je prístupná celá štruktúra z jedného spínača v strome cez koreň iba pomocou jednej operácie. Pole prostredia však môže mať tiež iné použitia. Prostredia môžu byť použité pre sledovanie tvorca štruktúry tým, že sa uloží identifikátor tvorca v prostredí všetkých vytvorených bunkových spínačov. Iný príklad je, že všetky bunkové spínače v strome nižšej úrovne, v ktorom všetky symboly, ktoré majú rovnaké meno, budú platiť pre rovnakú vec, môžu byť zoskupené tak, že majú rovnaké prostredie.

Preto keď je dané prostredie spínača, je možné nájsť koreňový spínač v tomto prostredí. Koreňový spínač prostredia môže byť vybavený konkrétnou značkou (napríklad "1") v poli WHERE vo svojej pamäťovej bunke. Uzlový spínač prostredia môže byť vybavený inou značkou (napríklad "0") v poli WHERE.

Typ.

Obsah poľa pamäťovej bunky TYPE je typ bunkového spínača. Táto zložka TYPE môže napríklad mať hodnoty: par, teda paralelné hodnoty a/alebo popisy, list, teda postupnosť hodnôt a/alebo popisov, unify, teda obsah hodnôt a/alebo popisov, ktoré

majú byť preskúšané, či predstavujú rovnakú hodnotu alebo nie, atď.

Jadrová bunka.

Jadrová bunka ako celok je aritmetické zariadenie pre štrukturálnu aritmetiku, ktoré obsahuje niekoľko jadrových registrov (viď obr.1). V registroch je uložený strom zoznamov. Každý zoznam obsahuje slová.

Jadrové registre obsahujú definíciu, ktorá je odvodená od objektového ukladania 1. Jadrové registre 3 sú pripojené k objektovému ukladaniu s usporiadaním zbernice 8 vrátane zberníc typu id, env, v_0 , v_1 , v_2 , v_3 , dostatočne širokej k prenosu obsahu v objektovej pamäťovej bunke. Jadrová bunka však môže obsahovať a zahrňovať až tri úrovne objektovej štruktúry. Existujú štyri prípady: 0, 1, 2 a 3 úroveň objektovej štruktúry môže byť uložená v jadrovej bunke.

Registre v jadrovej bunke.

Registre, ktoré môžu byť použité v príklade jadrovej bunky, sú uvedené na obrázkoch 4B až 4D, konfigurácia registrov, ktoré môžu byť použité v príklade jadrovej bunky je uvedená na obr. 4E.

Obr. 4B má znázorňovať, že register je zostavený z registrových buniek, každá bunka je schopná uložiť jeden bit informácie. Zpôsob, akým je register riadený, má znázorňovať to, že register sa rozkladá na rôznych rovinách v jadrovej bunke, pričom každá registrová bunka je umiestnená na jednej rovine.

Obr. 4C predstavuje register, ktorý sa rozkladá na všetkých rovinách v jadrovej bunke, to je plný register. Tento druh registra môže uchovávať identifikátor alebo hodnotu v registrovej bunke, ktorá sa nachádza v rovinách NUM a HEAD. Môže tiež uchovávať stav, uvedený vyššie, v registrových bunkách, ktoré sú v rovinách BOOL, TYPE, WHERE, LAZY a CLOS/SIMPLE.

Obr. 4D ukazuje register, ktorý sa rozkladá iba v rovinách NUM a HEAD jadrovej bunky, to je obmedzený register.

Obr.4E predstavuje možnú konfiguráciu registrov v jadrovej bunke. Jadrová bunka má bazové registre, ktoré sú najlepšie usporiadané do štvorca a nazývajú sa bazová registrová matica. Bazové registre majú hlavný rad pozdĺž jednej zo svojich strán a nazývajú sa hlavné registre. Stĺpce bazových registrov, kde každý má jeden hlavný register na spodnej strane, sa nazývajú vedľajšie registre. Jadrová bunka môže byť tiež vybavená registrom identifikátora a registrom prostredia. Rad pomocných registrov je umiestnených po strane matice bazových registrov.

V príklade jadrovej bunky všetky bazové registre okrem hlavných registrov môžu byť takého druhu, aký je ukázaný na obr.4D, teda obmedzené registre a zvyšok registrov na obr.4E môže byť taký, ako je ukázané na obr.4C, teda plné registre.

Pred uvedením podrobnejšieho popisu hardwarovej štruktúry jadrovej bunky bude uvedený krátky popis rôznych foriem ukladania dát vzhľadom k obr.5A až 5F a budú uvedené niektoré príklady jeho operácie vzhľadom k obrázkom 6H,7A až 7G a 8A až 8G.

Jednoduchá hodnota.

Ako je uvedené na obr. 5A, jednoduchá hodnota 25, ktorá je výsledkom redukcie, je v konkrétnom registri hlavných registrov.

Jednoúrovňová hodnota.

Cieľom je to, čo je načítané do jadrovej bunky a má byť redukované. Ako je ukázané na obr.5B, cieľ, zahrňujúci iba jednu štruktúru, ktorá je typicky spínač bez referencií na ostatné bunkové spínače, je uložený v hlavnom registri. Príklad ukazuje jednoduchú číselnú operáciu, to je sčítanie hodnôt 1, 2 a 3. Číselná inštrukcia (+) je uložená v prvom hlavnom registri a prvky, ktoré majú byť spracované, sú uložené v ostatných hlavných registroch.

Dvojúrovňová štruktúra.

Ako je ukázané na obr. 5C, strom vrátane dvojúrovňovej

Štruktúra môže mať svoj koreňový zoznam, ktorý je otcom, uložený horizontálne v hlavnom registri a zoznamy, ktoré sú synovia, horizontálne v báзовých registroch. V tomto príklade štruktúra, ktorá má reprezentáciu zoznamu ((1 2) (3 4)), je uložená v matici báзовého registra. Koreňový zoznam, to je 1 a 3, ktorý tvorí prvé prvky v podzozname, je uložený v hlavných registroch a synovské zoznamy, to je (1 2) a (3 4), sú uložené vo vedľajších registroch. Ďalšie príklady tohto druhu uloženia budú uvedené ďalej vo vzťahu k obr. 6.

Trojúrovňová štruktúra.

Ako je ukázané na obr. 5E, cieľový strom, ktorý zahŕňa trojúrovňovú štruktúru, má svoj koreň uložený v jednom z pomocných registrov a jeho jediný syn je uložený v hlavných registroch. Koreň na obr. 5D, ktorý je inštrukčným prevedením (Tr - Transpose) cieľového stromu, je uložený v jednom z pomocných registrov a jeho syn, ktorý je zoznamom (id1, id2, id7), je uložený v hlavných registroch. Každý prvok v tomto zozname je súčasne identifikátorom, označujúcim syna. Na obr. 2 sú títo synovia vertikálne načítaní do báзовých registrov, kde id1 je zamenené za zoznam, ktorý označuje, teda (1 2 3), id2 je zamenené za zoznam, ktorý označuje, teda (11 12 13) a kde id7 je zamenené za zoznam, ktorý označuje, teda (21 22 23).

Režim pipe line.

Ako je ukázané na obr. 5F, strom, ktorý je uložený v režime pipe line, je načítaný s cieľovým zoznamom v hlavných registroch a s otcom cieľa v pomocných registroch a má inštrukcie a prvky, ktoré majú byť spracované, uložené v obidvoch druhoch registrov. Režim pipe je používaný najradšej vtedy, keď sa redukujú číselné výrazy. Jednou výhodou je, že je možné okamžite ukladať medzivýsledky v jadrovej bunke namiesto v objektovom ukladaní.

Príklad 1.

Prvý príklad, ktorý je ukázaný na obr. 6A až 5H, je

zjednotenie paralelných hodnôt, daných ako redukovateľný spínač

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

Tento redukovateľný spínač má byť prepísaný ako paralelná štruktúra zjednotenia.

Obr.6A ukazuje počiatočný redukovateľný spínač. Obr.6B ukazuje, ako je tento redukovateľný spínač uložený v objektovom ukladaní. Pamäťové bunky, v ktorých sú uložené rôzne časti redukovateľného spínača, sú vyznačené na obr.6A. Spojenie medzi prvkovými spínačmi a bunkovými spínačmi sú vyznačené na obr.6B. Bunkový spínač, ktorý má identifikátor id_1 , má príznak cls a má kód typu unify v typovom poli a bunkové spínače, ktoré majú identifikátory id_2 , id_3 a id_4 , majú kód typu par vo svojich poliach typu. Bunkový spínač, ktorý má identifikátor id_1 , zahrňuje ako svoje prvé dva prvky spínača hodnota/popis, ktoré označujú bunkové spínače, ktoré majú identifikátor id_2 a id_4 . Tieto bunkové spínače majú príznak canon. Bunkový spínač, ktorý má identifikátor id_2 , má svoje prvé a tretie prvky spínača hodnota/popis vybavené diskretnými hodnotami, ktoré majú príznak discr, a svoj druhý prvok spínača hodnota/ popis, ktorý označuje bunkový spínač, ktorý má identifikátor id_3 a má tak príznak canon. Bunkový spínač, ktorý má identifikátor id_3 , má svoj prvý prvok spínača hodnota/popis vybavený pre typ integer a preto má príznak discr. Bunkový spínač, ktorý má identifikátor id_4 , má svoje prvé a tretie prvky spínača hodnota/popis vybavené diskretnými hodnotami, ktoré majú príznak discr a svoj druhý prvok spínača hodnota/popis, ktorý označuje bunkový spínač s identifikátorom id_3 a preto má príznak canon.

Ako je ukázané na obr.6C, obsah pamäťovej bunky s bunkovým spínačom s identifikátorom id_1 je najskôr načítané do jadrovej bunky tak, že najskôr umiestni svoj identifikátor v registri identifikátora ako id_1 vrátane kódu typu spínača unify a prvky hodnota/popis ako cieľ v hlavných registroch v prvom kroku operácie. Ako je toto v skutočnosti prevedené je ukázané na obr.13 a popísané ďalej.

Ako je ukázané na obr.6D, synovia ktorý majú identifikátory

id_2 a id_4 sú načítané vertikálne do základných registrov tak, že obsah ich prvého prvku hodnota/popis je umiestnený v hlavnom registri, označený svojim identifikátorom a zvyšok jeho prvkov hodnota/popis v registroch vo vertikálnych stĺpcoch. Kód typu par každého z týchto synov je tiež načítaný do hlavného registra. Kód typu je načítaný do registrových buniek, umiestnených v rovinách TYPE.

Ako je ukázané na obr. 6E, obsah základných registrov je transponovaný o 90°, takže obsah prvého vertikálneho stĺpca základných registrov je umiestnený v hlavných registroch a druhý vertikálny stĺpec je umiestnený v rade základných registrov paralelne s hlavnými registrami. Operácia transponovania je ukázaná na obr. 18 a je popísaná ďalej. Kódy typu par a unify, ktoré sú v registri identifikátora a v hlavných registroch, sú zamenené, čo sa vykonáva automaticky riadiacou jednotkou. Základné registre teraz obsahujú otca, ktorý má troch synov umiestnených v stĺpcoch. Každý syn je teraz načítaný späť do objektového ukladania s použitím inštrukcie make. Ako odpoveď z objektového ukladania sú k dispozícii identifikátory pre uložených synov a sú uložené v hlavných registroch. Je nutné si všimnúť, že riadiaca jednotka 6 sníma obsah zvlášť v registroch v rovinách CLOS/SIMPLE až TYPE a dáva inštrukcie, teda riadi spínače a hradlá podľa informácie, ktoré tu nájde. Synovia boli pomenovaní v postupnosti po id_1 a už obsadené názvy nie sú použité. Poradie názvov však nemá žiadny význam a môže byť preto ľubovoľný.

Ako je ukázané na obr. 6F, prvý syn dostane identifikátor id_2 druhý syn, ktorý obsahuje prvok spínača, ktorý okupuje identifikátor id_3 , dostane identifikátor id_4 , a tretí syn dostane identifikátor id_5 . Otec, ktorý má prvok spínačov spojený s bunkovým soľnačom, ktorý má identifikátory id_2 , id_4 , id_5 , si podržal svoj identifikátor id_1 a je potom uložený v objektovom ukladaní.

Obr. 6G ukazuje pamäťové bunky, ktoré majú uložený redukovateľný spínač

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

Redukovateľný spínač sám je ukázaný na obr. 6H. Obr. 6G a 6H

sú ukázané rovnakým spôsobom ako obr.6A a 6B a vysvetľujú preto sami seba.

V obr. 6G je tiež ukázané, že bunkovým spínačom, ktoré majú kód typu unify, bol daný v poli LAZY zápis exec a bunkovému spínaču s identifikátorom id_1 bol daný zápis wait, čo znamená, že bunkové spínače, ktoré sú označené exec, by mali byť vykonané pred bunkovým spínačom s identifikátorom id_1 aby sa znížil ich obsah na hodnoty. Spínač na obr.6H môže byť naskoršie načítaný späť do jadrovej bunky pre ďalšie spracovanie. Napríklad bunkový spínač s identifikátorom id_2 bude mať hodnotu 1, pretože hodnoty 1 a 1 vo svojich prvkoch hodnota/popis sú rovnaké a bunkový spínač s identifikátorom id_5 bude mať za výsledok nothing, pretože hodnoty 2 a 3 vo svojich prvkoch hodnota/popis nie sú rovnaké. Každé zjednotenie bude vykonané v číslicovej ALU, ktorá porovnáva hodnoty v komparátoroch a odovzdáva výsledok porovnania do riadiacej jednotky 6. Riadiaca jednotka potom použije svoje boolovské hradlové pole príslušne k odovzdaniu informácie v prvom hlavnom registri v jadrovej bunke. Keď je výsledkom redukcie kanonický popis alebo jednoduchá hodnota nothing (nič), je globálne distribuovaná do všetkých pamäťových polí v objektovom ukladaní tak, že každý nepriamy popis redukovaného spínača je nahradený výsledkom redukcie. To je uskutočnené pomocou operácie unify-id, ktorá je popísaná ďalej na obr.16.

Príklad 2.

Tento príklad je hardwarový inštrukčný list expansion, čo znamená, že bunkový spínač obsahuje vložený zoznam. Tento druh inštrukcie je pomocný krok v ostatných redukciách. Hardwarové inštrukčné rozšírenie zoznamu je ukázané na obr.19 a je popísané ďalej pri popise obr.19.

Počítač vykonáva redukciu ilustrovanej inštrukcie, ktorá sa nazýva ex. typ a ktorá môže byť ľubovoľným typom inštrukcie, ktorá zahrňuje hodnoty a zoznamy, ktoré majú tvar

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

Tvar je ukázaný na obr.7A a jeho bunkové spínače na

obr.7B.Obr.7A a 7B sú označené rovnakým spôsobom ako obr.6A a 6B a preto vysvetľujú sami seba.

Ako je ukázané na obr.7C,bunkový spínač s identifikátorom id_1 je načítaný do hlavných registrov jadrovej bunky, ktorá má svoj identifikátor a kód typu v registri identifikátora. Pretože obsah v druhom hlavnom registri je označený nepriamym prvkom open, bunkový spínač,ku ktorému je viazaný,je načítaný vertikálne v báзовom registri ako syn,ako je zrejme z obr.7D.

Hardwarový inštrukčný list expansion,uvedený podrobnejšie na obr.19 potom posunie diskretnu hodnotu 7 v treťom hlavnom registri do polohy vedľa id_4 v treťom báзовom stĺpci a posunie časť zoznamu v druhom stĺpci nad druhý hlavný register do tretieho stĺpca,pričom umiestni svoj najnižší prvok (hodnota 3) do tretieho hlavného registra a priradí mu kód typu list.Pretože obsah v druhom hlavnom registri je diskretna hodnota,má príznak discr.

Nové rozšírenie zoznamu je potom urobené umiestnením obsahu v treťom stĺpci nad hlavný register vo štvrtom stĺpci tak,že sa zapíše ako list (zoznam).Obsah tretieho hlavného registra,čo je diskretna hodnota,má príznak discr,ako je zrejme z obr. 7F.

Zoznam vo štvrtom stĺpci je potom uložený v objektovom ukladaní s použitím inštrukcie make.Je uložený v pamäťovej bunke,ktorá má identifikátor id_2 ,pretože sa dostala do kludu a napájanie identifikátora id_2 je vyslané späť do jadrovej bunky,aby bolo uložené vo štvrtom hlavnom registri,ako je zrejme z obr.7G.

Potom niektorý iný druh operácie je nastavený ako ex.type ešte predtým,ako sa výsledok redukcie načíta späť do objektového ukladania.

Príklad 3.

Má byť uskutočnená číselná inštrukcia.Číselná inštrukcia môže byť + , - , / , mod, atď.Po inštrukcii nasledujú argumenty. V tomto príklade má byť urobené sčítanie medzi číslami zo zoznamu.Počítač urobí redukciu aplikácie (apply),ktorá má funkciu

`apply(+ list(1 2))`

Aplikácia je uvedená na obr.8A a jej bunkové spínače na obr.8B. Obr.8A a 8B sú označené rovnakým spôsobom ako obr. 6A a 6B a preto vasvetľujú sami seba.

Ako je ukázané na obr.8C, bunkový spínač s identifikátorom id_1 je načítaný do hlavných registrov jadrovej bunky, ktorá má svoj identifikátor a kód typu v registri identifikátora. Číselná inštrukcia (+) je označená ako inštrukcia. Pretože obsah v druhom registri má príznak ako nepriamy prvok open, bunkový spínač, s ktorým sa spája, je načítaný ako syn vertikálne do báзовých registrov, ako je zrejmé z obr. 8D.

Potom je urobené rozšírenie zoznamu, diskretná hodnota sa označí príznakom v druhom hlavnom registri ako discr a označí sa hodnota rozšíreného zoznamu 2 ako list v poli kódu typu. To sa uskutoční, pretože počítač vykoná rovnaké operácie či má zoznam s identifikátorom id_2 dva, tri alebo štyri prvky. Pretože tu je iba jeden prvok v novom zozname, počítač zamení značku list za indikáciu, že hlavný register obsahuje hodnotu, ktorá je discr, ako je zrejmé z obr.8F.

Hlavný register obsahuje inštrukčnú značku (+) a dve diskretné hodnoty a to má za následok, že riadiaca jednotka priamo alebo napríklad cez informáciu, ktorá berie ohľad na uloženie inštrukcie v oblasti objektového ukladania, do ktorej nie je možné zapisovať a v ktorom sú uložené inštrukcie, bude riadiť číslicovú ALU tak, aby vykonala inštrukciu (sčítanie) a výsledok číselnej operácie odovzdala ako kanonickú hodnotu do prvého hlavného registra, ako je uvedené na obr.8G. Je nutné si všimnúť, že zápis apply v poli kódu typu je označenie, že sa vykonáva aplikácia funkcie. Výsledná hodnota, v tomto prípade jednoduchá hodnota 3, je potom globálne distribuovaná, aby sa vykonala výmena všetkých výskytov identifikátora id_1 za túto hodnotu.

Ako je uvedené, pevný počet registrov v registrovom zásobníku má uložený zoznam. Nepoužité registre sú konkrétne označené ako nepoužité. Stromový zoznam sa používa na riadenie a vykonanie výpočtu a ten je urobený za riadenia centrálnej

riadiacej jednotky 6 a pokiaľ bude vykonaná číselná aritmetická operácia, v spolupráci s číslicovou ALU 5. Výpočet je vykonaný prepísaním obsahu stromového zoznamu.

Zoznamy, ktoré majú až štyri prvky, môžu byť umiestnené v jednom bunkovom spínači. Je však možné pracovať s ľubovoľne dlhým zoznamom, potom ale taký zoznam musí byť rozdelený na niekoľko zoznamov, z ktorých každý bude mať dĺžku, ktorá je rovnaká alebo menšia ako maximálna dĺžka v jadrovej bunke. Jadrová bunka potom môže spracovať v danom okamžiku strom s určitou hĺbkou. So stromami, ktoré majú väčšiu hĺbkú, je možné pracovať, ale len časť stromu, ktorá má obmedzenú hĺbkú, je uložená v danom čase v jadrovej bunke, to znamená, že v danom čase je možné pracovať iba s časťou stromu.

Interfejs: objektové ukladanie - jadrová bunka.

Objektové ukladanie 1 a jadrová bunka 2 sú prepojené cez transformačný interfejs, ktorý vytvára adaptáciu signálu a zbernicu 8 zo šírkou spínača, to je usporiadanie zbernice, ktorá dokáže preniesť celý obsah bunkového spínača medzi jadrovou bunkou 2 a objektovým ukladáním 1. Usporiadanie zbernice 8 zahŕňa čiastočné usporiadanie zbernice OBVj, vrátane zbernice id, env, v_0 , v_1 , v_2 , v_3 , pripojených k registru rovín NUM v jadrových registroch 3 a čiastočné usporiadanie zbernice TAG, pripojenej k registrom rovín HEAD v jadrových registroch 3 a usporiadanie zbernice ATTRIBUTE vrátane zbernice z rovín LAZY, WHERE a TYPE k odpovedajúcemu poľu spínača v pamäťových bunkách. Interfejs 7 zosiluje a transformuje signály z jadrových registrov 3 cez usporiadanie zbernice so šírkou bunkového spínača, vhodné pre pamäťové bitové bunky v objektovom ukladaní. Zosiluje a transformuje tiež signály z objektového ukladania 1 v operáciach čítania, ktoré majú byť prispôsobené registrom jadrových buniek. Napriek tomu, že je uvedené umiestnenie interfejsu 7 v objektovom ukladaní 1, môže byť miesto toho umiestnený v jadrovej bunke. Interfejs pre bitovú bunku je uvedený na obr. 20 a je ďalej podrobne popísaný.

Operácie pamäťovej bunky.

Keď sa vrátíme k obr. 3, pamäťové bunky sú riadené zloženou číselnou informáciou z centrálnej jednotky 6, ktorá je distribuovaná do všetkých pamäťových buniek cez riadiacu zbernicu (nie je ukázaná), pripojenú k jadrovej bunke 2.

Bitové bunky 15 v každom prvku spínača môžu byť riadené zo svojej hlavy 16 tak, že bitové bunky vykonajú v danom okamžiku jednu z nasledujúcich operácií:

- rest kedy každá uložená bitová hodnota ponecháva uloženú hodnotu bitu
- read kedy sa prečíta uložená hodnota bitu v bitovej bunke
- write kedy sa zapíše bitová hodnota v bitových bunkách
- compare kedy sa porovnajú dáta slova, skladajúceho sa z bitových hodnôt, uložených v bitových bunkách, s iným dátovým slovom

Značka je nastaviteľná v závislosti na logických podmienkach, z ktorých sú všetky funkciou dát na druhej zbernici a a b, predchádzajúcej značke, výsledku porovnania v prípade operácie porovnania a riadiaceho signálu do pamäte z riadiacej jednotky 6.

Pretože sa používa asociatívne adresovanie, teda dáta sú vyhľadávané na základe uloženého obsahu miesto adresy, nie sú tu žiadne fyzické závislosti. Pamäť je asociatívna. Je možné použiť bunkový identifikátor, prostredie, typ, hodnotu informácie alebo ich kombináciu ako kľúč vyhľadávania.

Značkový bit alebo bity v prvkoch pamäťovej bunky, ktoré majú hodnotu CHOSEN, označujú, že prvky boli vybrané ako cieľ pre mechanizmus prístupu. Určité operácie vyhľadávania nastaví značky, čo je uvedené ďalej.

Tanto typ prístupu sa môže týkať jednej alebo niekoľkých pamäťových buniek. Ako už bolo uvedené, jedna z týchto operácií na viac bunkách je určitý druh operácie ukladania, ktorá môže uložiť identifikátor v mnohých zvolených prvkoch.

Vodič acc, zapojený k hlave 16, prepojuje všetky bitové

bunky v prvku. Všetky bitové bunky sú riadené signálmi na vodiči acc. Ostatné vodiče d a d , pripojené k bitovým bunkám, sú pripojené k odpovedajúcim bitovým bunkám v ostatných pamäťových bunkách v objektovom ukladaní, ako bude vysvetlené ďalej.

Dekodér priority.

Dekodér priority 11 obsahuje jednu sekciu pre každú pamäťovú bunku, kde každá sekcia má prvé spojenie pre REQUEST, na ktorom bitová hodnota "true" ("áno") predstavuje NEED a bitová hodnota "false" ("nie") NO NEED a druhé spojenie pre GRANT, kde bitová hodnota "true" predstavuje CHOSEN a "false" NOT CHOSEN.

Dekodér priority nastavuje maximálne jeden grant rovný CHOSEN. To môže byť zvolené tak, že prvá sekcia meraná v štruktúre, ktorá má REQUEST rovný NEED, bude CHOSEN. Príklad použitia dekodéru priority je popísaný ďalej na obr. 21.

V mnohých prípadoch protokol na pamäťovej zbernici komunikuje so všetkými pamäťovými bunkami. Avšak v niektorých prípadoch by malo byť usporiadaných a prečítaných niekoľko pamäťových buniek alebo prvkov spínača vo vnútri pamäťových buniek. To sa vykoná pomocou dekodéru priority 11. Z každej bunky ide signál REQUEST a dekodér priority 11 vráti signál GRANT.

Pamäť je riadená operáciami čítania, zápisu a vyhľadávania. Tieto operácie môžu byť kombinované do zložitejších operácií. Na niekoľkých vnútorných zberniciach a a b a na globálnych zberniciach môže byť vytvorená určitá aritmetika.

Vyhľadávanie sa vykonáva pomocou porovnávania s výsledkom FIT alebo DIFFERENT. Vyhľadávanie môže byť vykonávané nasledujúcimi spôsobmi:

- (1) Vyhľadávanie môže byť urobené jednotlivo pre každý prvok spínača a je nezávislé na zloženej informácii v ostatných prvkoch spínača.
- (2) Vyhľadávanie môže použiť porovnávanie so všetkými prvkami spínača v pamäťovej bunke. Výsledok musí byť FIT v každom prvku.
- (3) Vyhľadávanie môže byť urobené s použitím porovnávania so

všetkými prvkami spínača v pamäťovej bunke. Výsledok musí byť FIT minimálne v jednom z prvkov spínača.

Porovnanie môže byť vykonávané jedným z nasledujúcich spôsobov:

- (1) Porovnávajú sa dva bitové vzorce. Porovnanie má za výsledok iba FIT, kedy všetky odpovedajúce bity sú zhodné.
- (2) Porovnávajú sa dva bitové vzorce lebo iba jeden z nich je kódovaný tak, že jeden z bitov udáva, že informácia bitového vzorca odpovedá hodnote informácie v ARBITRARY alebo SPECIFIC. Pokiaľ pri porovnávaní jedna hodnota informácie odpovedá hodnote hodnote ARBITRARY, potom je výsledok FIT. Inak je výsledkom FIT iba keď sú dve špecifické informácie identické.

Funkcia zbernice realizuje pojem čítania alebo zápisu slova objektového ukladania. Bus je riadený funkciou access. Funkcia access je závislá na značke a/alebo hodnote druhých zberníc a a b.

Funkcia WIRED OR, realizovaná na jednobitovej zbernici, má zoznam boolovských výrazov. Vyhodnocuje OR medzi všetkými prvkami spínača v pamäťových bunkách. Fyzicky odpovedá vodiču, ktorý je nastavený poľom tranzistorov v prvkovej hlave 16.

Funkcia WIRED AND, realizovaná na jednobitovej zbernici b má zoznam boolovských výrazov. Vyhodnocuje logický AND medzi všetkými prvkami spínača v pamäťovej bunke. Fyzicky odpovedá vodiču, ktorý je nastavený poľom tranzistorov v prvkovej hlave 16. Funkcia priority dekodéru priority 11 má zoznam boolovských výrazov ako argument a výsledok v zozname boolovských výrazov odpovedajúcej veľkosti. Argument má prvý prvok, ktorý má najvyššiu prioritu. Po ňom nasledujú bity s nižšou prioritou. Prvý bit "áno" argumentu dá výsledok v odpovedajúcom pravdivom bite vo výsledku. Všetky ostatné bity sú "nie".

Prvky v pamäťových bunkách sú všeobecne použité pri prvom vyhľadávaní hodnôt v prvkoch a potom sa vykonajú operácie ako je čítanie a zápis v nájdenom prvku.

Hardwarová štruktúra jadrovej bunky.

Roviny NUM a HEAD majú svoje jadrové registrové bunky spojené k vodičom v zbernici OBJv, zbernici id, teda zbernici identifikátora a zbernicu ENV, teda zbernicu prostredia medzi rovinami 2 a interfejsom 7. Zbernica OBJv zahrňuje časti zbernice v0, v1, v2 a v3.

Pole jadrových registrových buniek je tak "rozrezané" kolmo na registre do rovín a registrové bunky, ktoré patria rovnakým rovinám NUM a HEAD, ale rozdielnym jadrovým registrom, sú prepojené navzájom spôsobom, uvedeným na obr.9.

V štruktúre minimálne rovín NUM a HEAD, uvedených na obr.9 je umiestnený štvorec registrových buniek v matici s $N \times N$ registrami $S_{0,0}$ až $S_{N-1,N-1}$, bázové registre.

Bázové registre sú vo väčšine aplikácií použité pre prvky spínania prechodného uloženia. Označovanie registrov bolo prísne rozdelené tak, že jeden druh označovania ako báza, hlavné a pomocné registre je použitý pokiaľ popis je smerovaný do skutočnej polohy registra a iný druh označovania ako synovský, cieľový a otcovský register sa používa vtedy, keď popis je smerovaný na funkciu registra.

V uvedenom príklade je $N=4$ a to by malo mať prednosť, ale môžu byť zvolené matice iných veľkostí (nie je ukázané). Najnižší rad registrových buniek $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ a $S_{3,0}$, ako je uvedené na obr.9, je pripojená na zbernicovú linku h0 a hlavné registrové bunky. Hlavné registrové bunky $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ a $S_{3,0}$ sú najčastejšie používané ako cieľové koreňové registre k číslicovej ALU 5 cez zbernicu NU, ktorá sa skladá z vodičov NU0 až NU3.

Bunka registra identifikátora ID je pripojená na linku id a bunka registra prostredia ENV je pripojená na linku env.

Zbernicová linka h_i sa dá pripojiť na zbernicovú linku vi spínačom SW_{vi} , kde i je číslo medzi 0 a 3. Zbernica, ktorá obsahuje zbernicové linky h0, h1, h2, h3 sa menuje OBJh, to je horizontálna objektová zbernica. Zbernica OBJh je medzi ostatnými vecami používaná pre vertikálne načítanie dát, teda v stĺpci registrov, v jadrovej bunke, ktorej dáta sú dané objektovým

ukladaním cez zbernicu OBJv. Toto je popísané ďalej - obr.15.

Zbernicové linky id , env , $v0$, $v1$, $v2$, $v3$ sa dajú pripojiť na zbernicovú linku spínačmi $SW_{id,ho}$, $SW_{env,ho}$, SW_{v0} , $SW_{vi,ho}$, $SW_{v2,no}$ a $SW_{v3,ho}$. Zbernica res , vrátane zbernicových liniek c_{id} , c_f , c_h a c_v , je pripojená k riadiacej jednotke 6 a môže byť použitá pre nastavenie registra s konštantou ako je nula. Zbernicová linka c_{id} je pripojená k bunke registra identifikátora a zbernicová linka c_f je pripojená k registrovým bunkám $F0$, $F1$, $F2$ a $F3$ a zbernicová linka c_h je pripojiteľná k zbernicovej linke $h0$ spínačom $SW_{vi,ho}$ a zbernicová linka vi spínačom $SW_{vi,cv}$, kde i je číslo medzi 0 a 3. Zbernica res s jej spínačmi môže byť v niektorých aplikáciách vynechaná (nie je uvedené).

Pomocné registre.

V dátovom strome je najvyššia úroveň, ktorá sa nazýva otec. Otec je niekedy uložený v pomocných registrových bunkách $F0$, $F1$, $F2$, $F3$, ktoré sú na obr. 9 umiestnené naľavo. Každý pomocný register v uvedenom príklade môže mať v sebe uložené jadrové slovo. Každá pomocná registrová bunka je pripojená na zbernicovú linku id a na jednu z liniek $h0$, $h1$, $h2$ alebo $h3$. Pomocné registrové bunky majú byť používané v menšine operácií, ktoré jadrová bunka môže vykonávať. Preto pomocný register môže byť v niektorých aplikáciách vynechaný (nie je uvedené). Je tiež možné mať jadrovú bunku, ktorá má viac ako jeden stĺpec pomocných registrov (nie je uvedené).

Ako je zrejmé z vyššie uvedeného, každý register je vybavený registrovou bunkou v niekoľkých rovinách 22, ktoré majú rovnaké umiestnenie v rovinách. Preto budú celé registre pomenované referenciami, použitými na obr.9, aj keď obr.9 uvádza iba jednu bunku, teda jeden bit z každého registra. Ako je zrejmé z obr.9. registre sú usporiadané v radoch a v stĺpcoch. Oblasť pomocných registrov $F0$, $F1$, $F2$, $F3$ je stĺpec a oblasti N bazových registrov $S_{0,0}$ až $S_{0,3}$, $S_{1,0}$ až $S_{1,3}$, $S_{2,0}$ až $S_{2,3}$ a $S_{3,0}$ až $S_{3,3}$ sú všetky v stĺpci a sú schopné uložiť syna.

Spoje medzi registrovými bunkami:

Medzi susednými bazovými bunkami v každej rovine je spojenie a to ako vertikálne tak i horizontálne. Spojenie, ktoré má pevne naprogramovanú hodnotu a ktoré v uvedenom príklade má hodnotu "nie", je tiež urobené ku každej registrovej bunke, ktorá je najviac na okraji v horizontálnom rade, otočenom k objektovému ukladaniu. V registrovej bunke je pripojená ku N svorke (North - sever) (viď obr. 10) a používa sa, keď sú vykonávané posuvy v smere Sever - Juh. Spojenie v diagonálnom smere medzi bazovými registrovými bunkami je nastaviteľné tak, že prehoditeľné polohy je možné zapojiť. To znamená, že bunka $S_{i,j}$, kde i je rozdielne od j , je pripojiteľná k bunke $S_{i,j}$. Každá bazová registrová bunka je pripojená k bazovej registrovej bunke najbližšie dolu vpravo, kde je bazová registrová bunka v takej polohe. Každá pomocná registrová bunka alebo registrová bunka identifikátora alebo prostredia je pripojená k jednej z rovín BOOL cez výstup ACC_{fx} , ACC_{id} , ACC_{env} a $ACC_{sx,y}$, kde x a y sú čísla medzi 0 a 3.

V nasledujúcom je popis generickej registrovej bunky (obr.10). Z generickej registrovej bunky sú odvodené príklady špecifických registrových buniek, to je bazové pomocné bunky a bunky identifikátora a prostredia (nie je uvedené). Čo sa týka ďalších podrobností o špecifickej štruktúre rôznych registrových buniek, odkazujeme na našu súbežnú US prihlášku vynálezu.

Generická registrová bunka.

Podľa obr.10 najlepšie použitie registrovej bunky zahrňuje dve interné zbernice a_R a b_R a centrálny interný register r_R . Zbernica a_R a b_R sú pripojené k niekoľkým prípojom zvonku registrovej bunky. Príklad na obr.10 ukazuje generickú registrovú bunku, ktorá má všetky možné prípoje von. Špecifická registrová bunka typicky nemá všetky prípoje, ktoré sú na obr.10 ukázané. V závislosti na umiestnení registrovej bunky ich chýba jeden alebo viac. Všetky vodiče medzi prepojenými svorkami sú zrejmé zo zapojenia na obr.9. Z obr.9 je tiež zrejmé, že nie všetky registrové bunky majú všetky vonkajšie prípoje, ktoré sú ukázané

na obr.10.Podrobný popis všetkých registrových buniek a ich prípojov nie je preto uvedený.

Zbernica a_R .

Zbernica a_R je pripojená na vertikálnu zbernicovú linku v_x , kde x je číslo medzi 0 a 3, cez spínač SW_{vi} a svorku V. Ďalej je pripojená na horizontálnu zbernicovú linku h_y , kde y je číslo medzi 0 a 3, cez spínač SW_{Hi} a svorku H, k registrovej bunke naľavo cez svorku W (West - západ) pripojená ku spínaču SW_E (East - východ) v susednej registrovej bunke a pokiaľ registrová bunka je hlavná registrová bunka, tiež k číslicovej aritmetickej jednotke lp priamo cez svorku NU. Zbernica a_R je tiež pripojená k registrovej bunke dolu cez svorku Da ku spínaču SW_{Db} , ktorý je v tejto registrovej bunke a tiež k registrovej bunke dolu cez svorku S (South - juh) ku spínaču SW_N (North), ktorý je v tejto registrovej bunke. Registrová bunka môže byť nastavená alebo resetovaná cez svorku C a spínač SW_C , ktorý je pripojený ku zbernici a_R . Zbernica a_R je pripojená tiež cez spínač SW_{a1} ku vstupu centrálného interného registra r_R a cez spínač SW_{ao} k jeho výstupu.

Zbernica b_R .

Zbernica b_R je pripojená k registrovej bunke napravo cez spínač SW_E a k svorke E a tiež k diagonále registrových buniek cez spínač SW_{Db} a svorku Db a tiež k vyššie uvedenej registrovej bunke cez spínač SW_N a svorku N. Zbernica b_R je tiež pripojená cez spínač SW_{b1} ku vstupu centrálného interného registra r_R a cez spínač SW_{bo} k jeho výstupu.

Centrálny interný register.

Centrálny interný register r_R obsahuje dva invertory $Q1$ a $Q2$, s výhodou invertory CMOS a ovládateľný spínač SW_Q medzi nimi. Kompletná registrová bunka obsahuje tiež zbernice a_R a b_R a spínače SW_{a1} , SW_{ao} , SW_{b1} , SW_{bo} a spínače, ktoré pripojujú bunku k vonkajšiemu okoliu. Výstup centrálného interného registra r_R je

pripojiteľný k horizontálnej a k vertikálnej zbernici cez spínač SV_{HO} a svorku H, respektíve spínač SV_{VO} a svorku V. V centrálnom internom registri r_R je uložený dynamický stav (vysvetlené ďalej).

Činnosť spínača.

Všetky ovládateľné spínače vo všetkých registrových bunkách v jadrovej bunke sú ovládateľné cez vodiče, pripojené k riadiacej jednotke, ktorá obsahuje hradlové pole ako napríklad PAL (pole s programovateľnou logikou). Hradlové pole používa informáciu, ktorá je uložená v jadrovej bunke k tomu, aby sa určilo, ktoré spínače majú ďalej byť otvorené a ktoré zatvorené. Operácie hradlového poľa sú synchronizované hodinami. Spínače sú obojsmerné, ale niektoré sú používané iba v jednom smere, napríklad vstupné a výstupné spínače SV_{Hi} a SV_{Ho} .

Komparačné zariadenie COMP.

Komparačné zariadenie obsahuje prvé NAND hradlo G1. Jeden vstup je zapojený k neinvertujúcemu vstupu invertora Q1 a ostatné ku vstupu invertora Q2. Zariadenie COMP tiež zahrňuje druhé NAND hradlo G2. Jeden vstup je pripojený k výstupu invertora Q1 a ostatné k výstupu invertora Q2. Výstupy hradiel G1 a G2 sú pripojené k jednovodičovej zbernici ACC, ktorá vedie k jednej z rovin BOOL. Obidve NAND hradlá môžu byť vybavené dvomi sériovo radenými tranzistormi MOS-FET, ktoré majú svoje sériovo radené kanály emitor/kolektor pripojené medzi zem a napäťový zdroj, pričom ich riadiace elektródy sú vstupy hradla NAND a kolektor najvyššieho tranzistora MOS-FET je výstup (nie je ukázané). Toto komparačné zariadenie COMP sa používa v priebehu asociatívneho vyhľadávania, to je keď prvok v jadrovej bunke bude porovnávaný s prvkom v objektovom ukladaní alebo inej časti jadrovej bunky. Potom prvok, ktorý má byť porovnávaný, je privedený na vstupy registrovej bunky, ktorá obsahuje porovnávaný prvok, čo bude popísané ďalej.

Invertory a spínače.

Invertory Q1 a Q2 môžu byť buď dva tranzistory MOS-FET obohateného typu alebo jeden tranzistor MOS-FET obohateného typu a jeden s vodivým kanálom alebo dva komplementárne tranzistory MOS-FET (nie je uvedené). Ovládateľné spínače v registrovej bunke môžu byť tvorené buď tranzistorom MOS-FET alebo dvomi komplementárnymi tranzistormi MOS-FET (nie je uvedené). Riadiaca jednotka 6 ovláda spínače pomocou riadiacich signálov. Spínač je možné ovládať ako jeho riadiacim signálom tak i jeho komplementárnym signálom, aby sa dosiahli rýchlejšie prenosy stavu.

Asociatívne vyhľadávanie a rovina(y) BOOL.

v priebehu asociatívneho vyhľadávania sa vykonáva porovnanie na prístupe zbernice WIRED-AND, ktorá ide do roviny BOOL. Dve hradlá AND Q1 a Q2 porovnávajú kľúčovú hodnotu, teda hodnotu, s ktorou má byť uložená hodnota porovnávaná, na vstupe Q1 a uloženú hodnotu na vstupe Q2. Behom tohoto porovnávania je kľúčová hodnota prenesená cez internú zbernicu a_R alebo b_R na Q1. Spínač SV_Q potom musí byť rozopnutý, teda otvorený. Pokiaľ daná hodnota, teda uložený kľúč, nie je zhodný s uloženou hodnotou, nabitá rovina BOOL bude vybitá cez NAND hradlo G1 a G2. Ak príde ku zhode rovina BOOL zostane nabitá.

Všetky zbernicové linky ACC v registri, jedna zbernicová linka ACC na registrovú bunku, môžu byť spojené paralelne a pripojené ku rovnakej zbernicovej linke v rovine BOOL. Ako alternatíva môžu byť zbernicové linky ACC všetkých registrových buniek v rovine NUM a HEAD pripojené ku zbernicovej linke v rovine BOOL, ktoré sú určené pre tieto roviny a všetky registrové bunky, vytvorené pre roviny ATTRIBUTE, môžu byť pripojené k samostatnej zbernicovej linke buď v rovnakej rovine BOOL alebo v druhej rovine BOOL, ktorá je určená pre roviny ATTRIBUTE. Ak sú vytvorené jedna alebo dve roviny BOOL a jedna alebo dve zbernicové linky, je záležitosť voľby a záleží na type riadiacich inštrukcií uložených v riadiacej jednotke 6. Či má byť viac rovín

ako dve je tiež predmetom vynálezu. Počet daných rovín BOOL určuje zrnitosť asociatívneho vyhľadávania, teda počet rôznych asociatívnych vyhľadávaní, ktoré je možné vykonať a do akého rozsahu sú vykonané, teda ktorých častí registra sa to týka. Preto je porovnanie vykonané súčasne pre časť registra, ktorý je pripojený ku rovnakej zbernicovej linke v rovine BOOL. Ak majú všetky NAND hradlá G1 a G2 rovnaký výstup (vysoká úroveň), potom je výsledok porovnania "súhlasí", inak je výsledok "nesúhlasí", pričom súhlasí znamená, že obe časti informácie sú zhodné. Roviny BOOL sú preto roviny pre zbernicové linky a je možné na ne pozerieť ako na skutočné alebo "myslené" roviny, teda že zbernicové linky môžu byť zapojené priamo k riadiacej jednotke 6.

Konfigurácia rovín ATTRIBUTE.

Roviny ATTRIBUTE môžu mať inú konfiguráciu ako roviny NUM a HEAD, môžu mať napríklad iba jeden rad bazových registrov namiesto matice bazových registrov (nie je uvedené). Ďalej môžu mať roviny ATTRIBUTE zvláštnu zbernicovú linku, ktorá môže byť pripojená ku všetkým alebo niekoľkým registrovým bunkám v rovine a ktoré idú na riadiacu jednotku 6, ktorá môže využiť informáciu na tejto zbernici k rozhodnutiu, aký druh redukcie sa má vykonať. Ďalej roviny ATTRIBUTE nemusia byť opatrené bunkami registra prostredia. Zbernicové linky v0, v1, v2 a v3 môžu byť privedené k ostatným vstupom interfejsu 7 namiesto k zbernicovým linkám, ktoré majú rovnaké označenie v rovinách NUM a HEAD a mohli by tak byť pripojené k ostatným častiam objektového ukladania 1, najlepšie k častiam lazy, where a type (viď obr.1). Ako alternatíva, zbernicové linky v0, v1 a v2 a v3 nemusia byť pripojené k objektovému ukladaniu 1. Namiesto toho zbernicová linka id v rovinách ATTRIBUTE môže byť použitá pre prenos informácie o stave (viď obr.1) z objektového ukladania, to znamená, že lazy, where a type v objektovom ukladaní sú ku zbernicovej linke id v odpovedajúcej rovine v jadrovej bunke.

Režim pohotovostného ukladania (standby)

Sľučku pohotovostného ukladania vytvárajú jedna alebo obidve sľučky, ktoré sú v registrovej bunke. Jedna sľučka je tvorená spínačom SW_{b0} , zbernicou b_R , spínačom SW_{b1} , invertorom $Q1$, spínačom SW_Q a invertorom $Q2$. Druhá sľučka je tvorená spínačom SW_{a0} , zbernicou a_R , spínačom SW_{a1} , invertorom $Q1$, spínačom SW_Q

a invertorom $Q2$. Keď sú spínače v jednej alebo obidvoch sľučkách uzavreté, signál môže postupovať cez invertory $Q1$ a $Q2$ a úroveň signálu na vstupe invertora $Q1$ a na výstupe invertora $Q2$ bude stabilná - to je spôsob, akým sú dáta ukladané do registrových buniek. Bunka ukladá dynamický stav.

Výstupný režim.

Vo výstupnom režime môže byť výstup $Q2$ prenesený na jednu zo zberníc a_R alebo b_R a odtiaľ je možné ovládať vhodné spínače tak, aby preniesli výstup na jednu alebo viac výstupných svoriek (N, S, E, W, atď.). Druhá zbernica a_R alebo b_R môže byť použitá v ľubovoľnom režime. Pokiaľ je spínač SW_Q rozopnutý, výstup invertora $Q2$ je stabilný, to je, že nemôže byť zmenený, pokiaľ spínač SW_Q bude zopnutý. Výstup invertora môže byť prenesený na zbernicu b_R cez spínač SW_{b0} keď je zopnutý, na zbernicu a_R cez spínač SW_{a0} keď je zopnutý. Informácia na zberniciach a_R a b_R môže byť prenesená na každú vonkajšiu stranu zberníc, ku ktorým je registrová bunka pripojená a to riadením spínača, ktorý je pripojený medzi registrovú bunku a vonkajšiu stranu danej zbernice, ako je ďalej uvedené na príklade.

Vstupný režim.

Vo vstupnom režime je jeden zo spínačov SW_{a1} alebo SW_{b1} zopnutý teda uzavretý. Preto je stav jednej zo svoriek (N, S, E, W atď) prenesený na lokálnu zbernicu a_R alebo b_R a odtiaľ do centrálného interného registra r_R .

Prenosy.

Ja možné prenášať dáta z ľubovoľného registra v jadrovej bunke cez svorkový spoj do inej registrovej bunky v jadrovej bunke v priebehu dvojfázového cyklu. V priebehu trojfázového cyklu je možný presun dvoch báзовých registrových buniek vo vertikálnom, horizontálnom alebo diagonálnom smere.

Spínač SV_Q je taktovaný priamo hlavnými hodinami a súčasne pre všetky bunky v registroch sa vykonávajú také presuny medzi invertormi Q_1 a Q_2 v celej jadrovej bunke. Zvyšok spínačov je riadený signálmi, ktoré sú odvodené od hlavných hodín, ale sú privedené v rôznych príslušných fázových intervaloch hlavných hodinových period. Hlavné hodiny sú použité ako referenčný signál pre všetky operácie jadrovej bunky.

Hodinový cyklus je rozdelený na hodinové fázy Q , a a/alebo b . Fáza Q je prvá natiahnuteľná fáza, to znamená, že centrálny interný register r_R je v režime pohotovostného ukladania - keď sú dáta stabilné. Fáza a sa používa behom presunu zo zbernice a_R a fáza b sa používa pri presune zo zbernice b_R .

Jednocestný presun, teda iba presun do alebo z registrovej bunky, sa vykonáva v dvojfázovom hodinovom cykle. Prvá fáza Q je pevná. V dvojfázovom hodinovom cykle sa pre presun používa fáza a alebo fáza b .

Dvojfázový presun, teda presun medzi registrovými bunkami pre výmenu ich vzájomného obsahu, sa vykonáva v trojfázovom hodinovom cykle. Fáza Q je pevná. V priebehu fázy a a b sa vykoná presun v rôznych smeroch.

Je nutné poznamenať, že hodinové cykly s viac ako tromi fázami sú predmetom vynálezu, napríklad dve fázy b .

Spínače SV_{a1} a SV_{b1} sú normálne zopnuté. Obe lokálne zbernice a_R a b_R potom udržujú uložený stav registrovej bunky. Keď interná zbernica a_R alebo b_R bude používaná pre zápis novej hodnoty pre uloženie, príslušný spínač SV_{a1} alebo SV_{b1} je riadený tak, aby bol otvorený. Prepnutie na jednu z externých zberníc, ako je vertikálna alebo horizontálna zbernica, je uzavreté behom krátkeho intervalu, ktorý je dostatočne dlhý pre prenos informácie z tejto zbernice na internú zbernicu.

Je tiež možné použiť sieť pre posúvanie, teda sieť medzi rôznymi registrovými bunkami vrátane spínačov, ktoré sú pripojené ku svorkám, aby sa vykonal prenos obsahu v registrovej bunke, sever N alebo juh S alebo západ W alebo východ E.

Príklad jednosmernej operácie presunu.

Obr. 11A ukazuje dve susedné bazové registrové bunky a dáta budú prenesené z ľavej (vysielače) do pravej (prijímače). Riadiace signály z riadiacej jednotky 6 riadia spínače. Obr. 11B ukazuje stav každého spínača, ktorý je ovplyvnený prenosom behom rôznych fází, nižšia hodnota predstavuje otvorený spínač vyššia hodnota predstavuje uzatvorený spínač. Skutočný prenos sa uskutoční vo fáze b. Prenos sa vykoná nasledujúcim spôsobom (rôzne kroky nižšie uvedené sú označené rovnakým číslom ako na obr. 11A a 11B):

0. Obvod je stabilný, SW_Q , SW_{a0} , SW_{a1} , SW_{b0} , SW_{b1} sú zopnuté, všetky ostatné spínače sú otvorené v oboch vysielateľoch (tento krok nie je na obr. 11 vyznačený, pretože sa týka všetkých spínačov). Tento stabilný stav odpovedá fáze 0 hodinového intervalu, keď spínač SW_Q vo vysieláči aj prijímači je otvorený.
1. V priebehu prvej fáze (fáze b) hodinového intervalu, kedy spínač SW_Q vo vysieláči aj prijímači je otvorený,
2. spínač SW_{a0} je otvorený a spínač SW_{b0} je uzatvorený vo vysieláči aj prijímači,
3. spínač SW_E medzi vysielateľom a prijímačom je uzatvorený,
4. spínač SW_{b1} vo vysieláči aj prijímači je otvorený,
5. spínač SW_{a1} vo vysieláči je otvorený a spínač SW_{a1} v prijímači je uzatvorený. To umožní, aby sa dáta dostali od interného registra vysieláča k internému registru prijímača.
6. V priebehu druhej fáze hodinového intervalu, kedy spínač SW_Q vo vysieláči aj prijímači je uzatvorený,
7. spínač SW_E medzi prijímačom a vysielateľom je otvorený,
8. spínače SW_{b0} a SW_{a0} sú uzatvorené ako prvé a potom aj spínače SW_{b1} a SW_{a1} vo vysieláči aj prijímači. Tým sa dostaneme späť do stabilného stavu, ktorý je už popísaný v kroku 0, teda vo

fáze 0.

Príklad dvojsmernej operácie.

Obr. 12A ukazuje dve susedné bazové registrové bunky a dáta dvoch rôznych bazových registrových buniek prehodené dvojcestnou operáciou. Riadiace signály z riadiacej jednotky 6 ovládajú spínače. Obr. 12B ukazuje stav každého spínača, ktorý je ovplyvnený presunom behom rôznych fáz, nižšia hodnota predstavuje otvorený spínač, vyššia hodnota predstavuje uzatvorený spínač. Obidve registrové bunky slúžia ako vysielateľ a prijímač, preto budú nazývané "bunka 1" a "bunka 2". Jeden prenos z bunky 2 do bunky 1 sa vykoná vo fáze a a prenos v opačnom smere z bunky 1 do bunky 2 sa vykoná vo fáze b. Rôzne kroky uvedené ďalej sú označené rovnakými číslami ako na obr. 12A a 12B. Prenos sa vykoná nasledujúcim spôsobom:

0. Obvod je stabilný, SW_Q , SW_{ao} , SW_{a1} , SW_{bo} , SW_{b1} sú uzatvorené, všetky ostatné spínače sú otvorené v obidvoch bunkách (tento krok nie je na obr. 12A označený, pretože sa týka všetkých spínačov). Tento stabilný režim odpovedá fáze 0 na obr. 12B,
1. V priebehu prvej fáze (fáza a) hodinového intervalu, keď spínač SW_Q v bunkách 1 a 2 je otvorený,
2. spínač SW_{ao} v bunkách 1 a 2 je uzatvorený a spínač SW_{bo} v bunkách 1 a 2 je uzatvorený,
3. spínač SW_E medzi bunkami je uzatvorený,
4. spínač SW_{a1} v bunkách 1 a 2 je otvorený,
5. spínač SW_{b1} v bunke 1 je uzatvorený a spínač SW_{b1} v bunke 2 je otvorený. To umožní, aby sa dáta dostali z bunky 2 do bunky 1.

V priebehu druhej fáze (fáza b) hodinového intervalu, kedy spínač SW_Q je stále otvorený,

6. spínač SW_{ao} v bunkách 1 a 2 je otvorený a spínač SW_{bo} v bunkách 1 a 2 je uzatvorený,
7. spínač SW_{b1} v bunkách 1 a 2 je otvorený a
8. spínač SW_{a1} v bunke 1 je otvorený a spínač SW_{a1} v bunke 2 je uzatvorený, čo umožní prenos dát z bunky 1 do bunky 2.
9. V priebehu tretej fáze (fáza 0) hodinového intervalu, kedy

- spínač SW_Q v oboch bunkách 1 a 2 je uzatvorený,
10. spínač SW_E v bunke 1 je otvorený a
11. spínače SW_{bo} a SW_{ao} sú uzatvorené ako prvé a spínače SW_{b1} a SW_{a1} v oboch bunkách po nich. Tým sa dostaneme späť do stabilného režimu, ktorý je popísaný vyššie v kroku 0, teda fáza 0.

Riadiace signály pre spínače SW_{ao} a SW_{bo} .

V priebehu fázy 0 sú signály implicitne aktívne, teda uzatvorené. Všetky lokálne zbernice potom udržujú uložený stav. Zbernica, použitá pre vstup je riadená nastavením riadiaceho signálu do stavu vypnuté, teda otvorené pre spínače SW_Q a spínač SW_x0 , kde x je a alebo b. V priebehu operácie vstupu môže byť niekoľko zberníc na krátku dobu skratované niektorými svorkami (E, V, D, H, atď.). Po chvíli sa zbernice dostanú na správnu hodnotu.

Medzi tylovou hranou riadiaceho signálu pre spínač SW_Q a tylovou hranou riadiaceho signálu pre spínač SW_{x0} (x je a alebo b) je časové oneskorenie. Pokiaľ je krátke, nepríde k žiadnym problémom. Keď sa však tento čas dostane do oblastí ms, zbernica x (x je a alebo b) môže stratiť svoj dynamický stav.

Medzi nábežnou hranou riadiaceho signálu pre spínač SW_{x0} a nábežnou hranou riadiaceho signálu pre spínač SW_{x1} (x je a alebo b) je časové oneskorenie. Pokiaľ je toto oneskorenie záporné, môže byť na lokálnej zbernici x prenesená chybná hodnota z invertora Q2 do invertora Q1. Preto sa používa kladné oneskorenie.

Riadiace signály pre spínače SW_E , SW_V , SW_D , SW_H atď.

Spínače sú normálne vypnuté teda otvorené. Všetky lokálne zbernice sú potom izolované. Zbernica, ktorá sa používa pre vstup alebo výstup, je riadená nastavením riadiaceho signálu pre svorkový spínač, ktorý je k nemu pripojený, do stavu zapnuté, to znamená zopnutím spínača. Počas tejto operácie môže byť niektorými spínačmi (SW_E , SW_V , SW_D , SW_H atď.) niekoľko zberníc na krátku dobu skratovaných. Po krátkej chvíli budú na zbernici správne hodnoty.

Od tylovej hrany riadiaceho signálu pre spínač SW_Q do nábežnej hrany riadiaceho signálu pre spínač SW_Z (Z začína H, D, N,

V, E atď., to je ktorákoľvek svorka, pripojená na internú zbernicu a_R alebo b_R , na ktorej je spínač). Pokiaľ je záporné, hodnota lokálnej zbernice x (je a alebo b) sa môže meniť. Hodnota registra potom môže byť nastavená.

Od nábežnej hrany riadiaceho signálu pre spínač SW_Z (Z začína H, D, N, V, E atď., to je ktorákoľvek svorka, pripojená k internej zbernici a_R alebo b_R , na ktorej je spínač) k tylovej hrane riadiaceho signálu pre spínač SW_{x1} . Pokiaľ bude toto oneskorenie záporné, hodnota nemôže postúpiť ďalej na vstup. Preto sa používa kladné oneskorenie.

Od nábežnej hrany riadiaceho signálu pre spínač SW_{x1} do tylovej hrany riadiaceho signálu pre spínač SW_Z je oneskorenie. Ak by bolo záporné, lokálna zbernica sa môže meniť a register môže byť nastavený na chybnú hodnotu. Preto sa používa kladné oneskorenie.

Riadiace signály pre spínače SW_{a1} a SW_{b1} .

Signály sú implicitne nastavené do stavu logickej jednotky vo fáze Q . Od nábežnej hrany riadiaceho signálu pre spínač SW_Q však musí byť malé oneskorenie do nábežnej hrany riadiaceho signálu pre spínače SW_{a1} a SW_{b1} .

Pokiaľ by toto oneskorenie bolo záporné, hodnota na vstupe invertora $Q2$ nemusí byť schopná postúpiť ďalej na zbernicu x_R (x je a alebo b). Preto sa používa kladné oneskorenie.

Výpočet jadrovej bunky.

V jednom strojovom cykle sa vykonávajú typické inštrukcie zoznamu.

Ako už bolo uvedené, jadrová bunka vykonáva štruktúrálnu aritmetiku. Všetky kroky vykonávajú jadrové registre s využitím inštrukcií v zozname, ktoré tento obsahuje. Príklady inštrukcií sú nasledujúce:

Dĺžka	Vypočítava sa dĺžka cieľa
map	Funkcia je aplikovaná na prvky zoznamu. Ak zoznam obsahuje vložené zoznamy, inštrukcia je tiež aplikovaná

na prvky týchto vložených zoznamov. (Inštrukčná mapa bude vysvetlená naskoršie.)

- filter Funkcia je aplikovaná a filtruje prvky zoznamu. Filter je tiež aplikovaný na vložené zoznamy, pokiaľ sú.
- join Všetky prvky sú prepísané na prvky vloženého zoznamu. Inštrukcia je tiež aplikovaná na vložené zoznamy pokiaľ sú.
- transpose Transponuje sa malá matica. Pokiaľ obsahuje prvky zoznamu, sú prehodené. Pracuje sa i s vloženými zoznamami. (Inštrukcia transponovania bude vysvetlená neskôr.)
- atd.

Ukladanie jadrovej bunky.

V jadrovej bunke je uložené

- * Cieľ, ktorý má byť redukovaný v niekoľkých registroch, najlepšie v báзовých registroch
- * V niektorých prípadoch, napríklad keď sa redukuje štruktúra stromovej úrovne, koreň cieľa, najlepšie v pomocných registroch a zvyšok štruktúry v matici báзовých registrov.

Objektové ukladanie 1 môže mať uloženú iba jednoúrovňovú štruktúru, ale jadrové registre 3 môžu obsahovať najviac trojúrovňovú objektovú štruktúru. Sú štyri prípady pre prechodné ukladanie v jadrových bunkách a to ukladanie objektov s úrovňou 0, 1, 2 alebo 3. Pokiaľ má byť uložený objekt s trojúrovňovou štruktúrou, uloží sa iba jeho koreň a jedna z jeho vetiev.

Jednoduchý strom, teda jednoduchá hodnota (objekt s úrovňou 0) sa uloží v prvom hlavnom registri.

Strom, ktorý obsahuje iba jednu úroveň, je uložený v hlavných registroch.

Strom, ktorý obsahuje dve úrovne, môže mať svoj koreňový zoznam, ktorý je otec, uložený horizontálne v hlavných registroch a zoznamy, ktoré sú synovia, vertikálne v báзовých registroch. Ako alternatíva - koreň môže byť uložený v pomocných registroch a jeden z jeho synov v hlavných registroch. Je nutné tiež poznamenať, že riadiaca jednotka 6 môže zvoliť jednu alebo druhú

alternatívu v závislosti na vykonávanej práci.

Strom, ktorý zahrňuje tri úrovne, má svoj koreňový zoznam uložený v jednom z pomocných registrov a jeden z jeho dvojúrovňových synov je uložený v matici bazových registrov.

Preto koreňový zoznam cieľového stromu je najlepšie uložený na rôznych miestach v registroch v jadrovej bunke v závislosti na úrovni stromovej štruktúry a operácii, ktorá sa má vykonávať.

Koreň cieľového stromu je spínač redukovateľného druhu ako unify. V aplikácii funkcie (apply ...) prvý prvok je inštrukcia alebo identifikátor nepriamo určujúci štruktúru spínača, ktorý sa používa ako definícia funkcie a zvyšok prvkov sú argumenty definície inštrukcií/funkcií.

Riadenie ukladania jadrovej bunky.

Informácia uložená v jadrových registroch je odvodená od informácie v objektovom ukladaní 1. Informácia v jadrových registroch je uložená nasledujúcim spôsobom:

Jadrové registre v rovinách HEAD a NUM sú pripojené na zbernicu objektového ukladania OBJ, prístupovú zbernicu ACC, zbernicu res a zbernicu NU číslícovej ALU. Uložený stav sa skladá z uloženého stavu dvoch jednoduchých registrov ID a ENV, pomocných registrov F0 až F3 a bazových registrov $S_{0,0}$ až $S_{3,3}$. Jadrové registre v rovinách ATTRIBUTE sú pripojené k objektovému ukladaniu podobným spôsobom, ale k rozdielnym častiam objektového ukladania 1, najlepšie k častiam lazy, where a type.

Riadiace slovo jadrových registrov sa skladá z riadiacich slov k prepínačom SW_{vi} , $SW_{vi,cv}$, kde i je číslo medzi 0 až 3, $SW_{id,h0}$, $SW_{ch,h0}$, $SW_{env,h0}$, $SW_{v1,h0}$, $SW_{v2,ho}$, a $SW_{v3,ho}$, SW_{vio} , SW_{v1} , SW_{v2} a SW_{v3} jednoduchým registrom ID a ENV, pomocným registrom F0 až F3 a bazovým registrom $S_{0,0}$ až $S_{3,3}$.

Riadiace slová sú prenesené cez niekoľko riadiacich vodičov, ktoré sú pripojené k riadiacej jednotke 6. Riadiace vodiče môžu byť

dvojfázové riadiace párové vodiče alebo jednofázové jednoduché riadiace vodiče, čo závisí od toho, aký druh prepínačov sa používa.

Riadiace slovo pre každú báзовú registrovú bunku obsahuje jednu spoločnú časť a časti, ktoré sú individuálne pre každý báзовý register. Spoločná časť riadi spínače SW_{a0} , SW_{b0} a SW_Q jadrovej bunky (obr. 10). Je však nutné poznamenať, že to, čo je tu popísané, je nutné považovať iba za príklad a sú možné aj iné príklady.

Príklady operácie jadrovej bunky.

Obr. 13 až 19 sú odvodené od obr. 9. Referencie na obr. 9 platia tiež pre obr. 13 až 19. Z dôvodu čitateľnosti však bola väčšina referencií vynechaná. Tiež v popisoch obr. 13 až 19 majú významy registrových buniek znamenať celý register, ktorý sa rozprestiera cez roviny v jadrovej bunke.

1. Prístup k objektovému ukladaniu.

Inštrukcia `mpx_mv`:

Operáciou objektového ukladania `mpx_mv` sa prečíta objektové ukladanie 1 a niektoré jadrové registre sa nastavujú. Adresovaný objekt je prenesený cez zbernice v0, v1, v2, v3 do hlavných registrov $S_{0,0}$, $S_{1,0}$, $S_{2,0}$, $S_{3,0}$ cez zbernicu `id` do hlavných registrov `ID` a cez zbernicu `env` do registra `ENV`, ako je znázornené na obr. 132 silnými čiarami, ktoré majú šípky smerom k registrovým bunkám, do ktorých sa prenos vykonáva. Súčasne je starý obsah hlavných registrov uložený v objektovom ukladaní 1 ako spínač. Preto inštrukcia `mpx_mv` ukladá súčasný spínač jadrovej bunky v objektovom ukladaní a načíta ďalší spínač objektového ukladania, ktorý má byť vykonaný v jadrovej bunke.

Inštrukcia `fetch`:

Obr. 14 a 15 znázorňujú situáciu, kedy je v jednom hlavnom registri uložený identifikátor a tento identifikátor má byť zamenený za informáciu, ktorú označuje. Identifikátor, uložený napríklad v $S_{2,2}$ (obr.14) je privedený k objektovému ukladaniu 6, objektové ukladanie nájde identifikátor a obsah, ktorý označuje, tento obsah je privedený na zbernicové linky v_0 až v_3 a nakoniec načítaný do vertikálnych stĺpcov v báзовých registroch, napríklad $S_{2,0}$ až $S_{2,3}$ (obr.15).

Táto operácia sa spustí prenosom identifikátora v báзовom registri $S_{2,0}$ do vertikálnej zbernice id cez zbernicu h0 a spínač $id_{1d,h0}$ (obr.14). Uložená hodnota môže byť prenesená z ľubovoľného iného registra podobným spôsobom.

Operácia pokračuje (obr.15) načítaním hodnôt, ktoré sú privedené objektovým ukladáním 1 na zbernicové linky v0, v1, v2, v3 do príslušných registrov, čo v danom príklade sú registre $S_{2,0}$, $S_{2,1}$, $S_{2,2}$, $S_{2,3}$ prenosom hodnôt cez spínače SW_{v0} , SW_{v1} , SW_{v2} , SW_{v3} a zbernice h0, h1, h2, h3.

Operácia objektového ukladania make a unify_id môže byť použitá, keď obsah v jadrovej bunke má byť uložený do objektového ukladania 1.

Inštrukcia make:

V prvom kroku operácie make je obsah daných registrov prenesený ako je uvedené na obr.15, ale v opačnom smere. Operácia tiež prenáša obsah registra prostredia. Vykoná sa asociatívne vyhľadávanie tak, aby sa našiel objekt s rovnakou informáciou, uloženou ako informácia, ktorá je v jadrovej bunke. Ak sa objekt nájde, vráti sa identifikátor, ktorý označuje objekt, keď sa objekt nenájde, vráti sa identifikátor unused (nepoužitý). V oboch prípadoch je identifikátor prenesený z objektového ukladania do registra v jadrovej bunke pomocou zbernicovej linky id. Ako alternatíva môže byť identifikátor prenesený do hlavného registra v danom stĺpci registrov. Preto sa vytvorí asociácia medzi obsahom jadrovej bunky a identifikátorom.

Inštrukcia `unify_id`:

Inštrukcia `unify_id` je uvedená na obr.16 a vykonáva distribúciu identifikátora z jedného z registrov, napríklad $S_{2,0}$, do všetkých vertikálnych zberníc `id`, `env`, `v0`, `v1`, `v2`, `v3` tak, že pripojí danú registrovú bunku na horizontálnu zbernicu `h0` cez spínače $SW_{id,h0}$, $SW_{env,h0}$, SW_{v0} , $SW_{v1,h0}$ atď. To je operácia, ktorá môže byť použitá pri vykonávaní asociatívneho vyhľadávania a nahradenia, ktoré napríklad môže vyvolať vyhľadávanie výskytu identifikátora a nahradenie nájdeného výskytu tohto identifikátora novou redukovanou jednoduchou hodnotou.

Operácia, ktorá je podobná `unify_id` môže vo svojom prvom kroku použiť make k tomu, aby sa získal jednotný identifikátor pre obsah jadrovej bunky a vo svojom druhom kroku odovzdať obsah jadrovej bunky na zbernicové linky, ktoré sú pripojené k objektovému ukladaniu, aby sa objektovým ukladáním uložil identifikátor a obsah, ktorý onačuje.

Príklad operácie `unify_id` je uvedený v prílohe 1, kde je ukázaný obsah jadrovej bunky a stav spínača pre fáze a a b.

2. Číslíkové redukcie.

V priebehu číslíkových redukcii objekt, ktorý má byť redukovaný, je umiestnený do hlavných registrov. Obecné celé cieľ sa zúčastní redukcie. Hlavný register $S_{0,0}$ typicky má v sebe uložený inštrukčný kód, ktorý je odlišný dvojité vzorec pre rôzne inštrukcie. Registre $S_{1,0}$ a $S_{2,0}$ sú použité pre dvojčlenné operácie, teda operácie s dvomi operandami a register $S_{1,0}$ je použitý pre jednočlenné operácie, teda operácie s jedným operandom. Obecné sú koncové registre použité v tvare zoznamu, ich obsah je preto pre redukciu vytlačený doľava.

Podstatná aritmetika sa potom vykonáva medzi registrami $S_{1,0}$ a $S_{2,0}$ cieľa. Hlavná sčítačka ALU je pripojená k týmto dvom registrom. Pre dodatkové účely sa môžu použiť ostatné registre v inštrukciách ako `mul`, `div` a `mod`.

Môžu byť použité nasledujúce typy inštrukcií:

Jednočlenné inštrukcie.

V registri $S_{0,0}$ T je uložená inštrukcia a v registri $S_{1,0}$ je uložený operand. Registre $S_{2,0}$ a $S_{3,0}$ nie sú používané. Výsledok z číslícovej ALU je vrátený do všetkých hlavných registrov. V prípade, kedy nejde o pipe-line, je uložený v registri $S_{1,0}$. V prípade pipe-line je okamžite uložený v pomocnom alebo bázovom registri.

Dvojčlenné inštrukcie.

V registri $S_{0,0}$ je uložená inštrukcia a v registri $S_{1,0}$ je uložený operand. Register $S_{3,0}$ nie je používaný. Výsledok je vrátený do všetkých hlavných registrov. V prípade, že nejde o pipe-line, je uložený v registri $S_{1,0}$. V prípade pipe-line je okamžite uložený v pomocnom alebo v bázovom registri.

Inštrukcie mul, div, mod

V registri $S_{0,0}$ je uložená inštrukcia a v registroch $S_{1,0}$, $S_{2,0}$ sú uložené operandy. Konečný výsledok je uložený v registri $S_{1,0}$.

Redukcie unify.

Redukcie unify používajú číslícovú ALU na komparáciu obsahu obsahu registra $S_{0,0}$ s obsahom $S_{1,0}$. Ostatné hlavné registre môžu byť pri vykonávaní zjednotenia tiež použité. Príznakové slová, ktoré sú uložené v rovinách HEAD registrov sú použité s výsledkom komparácie k vyhodnoteniu ďalšej činnosti.

Inštrukcie mul, div a mod vykonávajú svoje vnútorné sľučky úplne vo vnútri číslícovej aritmetickej jednotky. Vypočítané hodnoty medzivýsledkov môžu byť ukladané dynamicky na vodičoch medzi číslícovou aritmetickou jednotkou a hlavnými registrami v jadrovej bunke, teda na zbernici NU.

3. Redukcie štruktúry.

Počas redukcie štruktúry objekt, ktorý má byť redukovaný, teda cieľ, je umiestnený v hlavných registroch. Obecne niektoré alebo všetky bazové registre sa zúčastnia redukcie. Obyčajne v hlavnom registri $S_{0,0}$ je uložená inštrukcia, čo je iný dvojité vzorec pre rôzne inštrukcie.

Inštrukcia mapy má funkciu f a list($e_1, \dots, e_n$) ako argumenty a aplikuje funkciu na každý prvok zoznamu. Inštrukcia vracia zoznam ($f_{e_1}, \dots, f_{e_n}$) výsledkov každej aplikácie funkcie, kde f_{e_1} predstavuje výsledok aplikácie f na e_1 .

Inštrukcia mapy.

Formát: (map f list).

Inštrukcia mapy je načítaná do pomocného registra F0. Funkcia, ktorá má byť použitá, je načítaná do pomocného registra F1. Zoznam je načítaný v hlavných registroch $S_{0,0}$ až $S_{3,0}$. Ako je uvedené na obr.17A, prvky uložené v hlavných registroch sú prenesené v dvoch krokoch do matice bazových registrov, teda obsah v registri $S_{x,0}$ je prenesený do registra $S_{x,2}$, kde x je číslo medzi 0 až 3. Prenos je vykonaný s použitím vertikálnych zbernicových liniek v_0 až v_3 . Ako je uvedené na obr.17B, obsah v registroch F0 a F1 je potom vysielaný horizontálne do bazových registrov, teda obsah F0 je prekopírovaný do $S_{0,0}$ až $S_{3,0}$ a obsah F1 je prekopírovaný do $S_{0,1}$ až $S_{3,1}$. Pokiaľ je prvok jednoduchá hodnota (nie je to zoznam), obsah registra, v ktorom je umiestnená, napríklad $S_{1,1}$, je posunutý jeden krok smerom dole. Funkcia, ktorá má byť aplikovaná, je teraz umiestnená v hlavnom registri napríklad $S_{1,0}$ a prvok, na ktorý má byť funkcia aplikovaná, je teraz umiestnený v registri nad týmto hlavným registrom, napríklad v $S_{1,1}$. Pokiaľ je prvok zoznam, nevykoná sa žiadny posuv v tomto stĺpci registrov. Na obr.17C sa predpokladá, že e_1 , e_2 a e_3 predstavujú jednoduché hodnoty a že e_4 predstavuje zoznam. Každý stĺpec v matici bazového registra je potom uložený ako spínač v objektovom ukladaní. Potom každý z týchto uložených spínačov je načítaný do jadrovej bunky pre pokračujúce

spracovanie. Ak uložený spínač obsahoval jednoduchú hodnotu, je načítaný do jadrovej bunky normálnym spôsobom, teda f je uložené v $S_{0,0}$ a e_i je uložené v $S_{1,0}$, ako je uvedené na obr.17D. Ak na druhej strane uložený spínač obsahoval zoznam, je načítaný tak, ako už bolo uvedené na obr.17A, ale s e_1 ako prvý prvok v zozname, ktorý je predstavovaný e_4 , e_2 druhý prvok v e_4 atď. To umožní, aby inštrukcia mapy pracovala rekurzívne vo vnútri vloženého zoznamu. Preto inštrukcia mapy, ktorá má dve úrovne štruktúry

$$(\text{map}, f (e_1, \dots, e_n))$$

je prepísaná do

$$((f, e_1, \dots, (f, e_n)),$$

čo je po vykonaní prepísané do jednoúrovňovej štruktúry

$$(f_{e_1}, \dots, f_{e_n}),$$

kde f_{e_1} predstavuje výsledok pri aplikácii f na e_1 .

Inštrukcia mapy, ktorá má tri (alebo viac) úrovne štruktúry

$$(\text{map}, f, \text{par}(e_1, \dots, (e_k, \dots, e_m), \dots, e_n)),$$

kde $(e_k, \dots, e_m)$ je vložený zoznam, je prepísaná na

$$\text{par}((f, e_1), \dots, (\text{map}, f, (e_k, \dots, e_m)), \dots, (f, \dots, e_n))$$

ako medzivýsledok kroku a potom na

$$\text{par}((f, e_1), \dots, ((f, e_k), \dots, (f, e_m)), \dots, (f, e_n)),$$

čo je po vykonaní prepísané na dvojúrovňovú štruktúru

$$\text{par}(f_{e_1}, \dots, (f_{e_k}, \dots, f_{e_m}), \dots, f_{e_n}),$$

kde f_{e_1} predstavuje výsledok po aplikácii f na e_1 a kde $(f_{e_k}, \dots, f_{e_m})$ je vložený zoznam.

Potom funkcia f je rekurzívne aplikovaná na všetky prvky zoznamu argumentov.

V ďalšom je popísaný ilustratívny príklad, ako sa jadrová bunka pretvorí a vykoná sa inštrukcia mapy. Skratka reg sa používa namiesto "register", ident namiesto "identifikátor" a storage namiesto "object storage" (objektové ukladanie), aby popis zostal krátky.

Príklad inštrukcie:

$$(\text{map } f (-1 -2 (-7 -8))),$$

kde f je definované ako $f(x) = \text{abs}(x) + 1$. Strojová reprezentácia, ktorá používa strojové identifikátory, môže byť:

```
id1: (map f id2)
```

```
id2: (-1 -2 id3)
```

```
id3: (-7 -8)
```

kde ident id1 označuje spínač, ktorý obsahuje (map f id2)
-structure atď.

Ďalej i je číslo medzi 0 a 3. Sú vykonané nasledujúce kroky:

Krok 1: mapa je uložená v reg F0, f v reg F1 a ident id2 v reg $S_{0,0}$.

Krok 2: ident id2 je rozšírené, teda reg $S_{0,0}$ obsahuje -1, reg $S_{1,0}$ obsahuje -2 a reg $S_{2,0}$ obsahuje ident id3

Krok 3: Obsah reg $S_{1,0}$ je prenesený do reg $S_{1,2}$. Registre, ktoré majú označenie unused (nepoužité), zostávajú nedotknuté.

Krok 4: map a f sú vysielané horizontálne, teda reg $S_{1,1}$ obsahuje f a reg $S_{1,0}$ obsahuje mapu. Registre, ktoré majú označenie unused (nepoužité), zostávajú nedotknuté.

Krok 5: Stĺpce s jednoduchou hodnotou vo svojom reg $S_{1,2}$ sú stlačené o jeden krok nižšie, teda reg $S_{0,1}$ obsahuje -1 a reg $S_{0,0}$ obsahuje f a reg $S_{1,1}$ obsahuje -2 a reg $S_{1,2}$ obsahuje f, tretí stĺpec zostáva nedotknutý.

Krok 6: Každý stĺpec v matici bázo­vého registra je uložený ako:

```
id1: (id6 id7 id8)
```

```
id6: (f - 1)
```

```
id7: (f - 2)
```

```
id8: (map f id3)
```

Krok 7: Spínač označený ako ident id6 je načítaný do hlavných registrov, f do reg $S_{0,0}$ a -1 do reg $S_{1,0}$.

Krok 8: Funkcia, teda $f(x) = \text{abs}(x) + 1$ je aplikovaná na argument s výsledkom 2, ktorý je uložený v reg $S_{0,0}$.

Krok 9: Vykonáva sa asociatívne vyhľadávanie pre ident id6 v uložení a všetky výskyty ident id6 sú nahradené dvoj­kou:

```
id1: (2 id7 id8)
```

```
id7: (f -2)
```

```
id8: map f id3)
```

Krok 10: Kroky 7 až 9 sú vykonané pre ident id7 a s výsledkom 3.

Uloženie:

id1: (2 3 id8)

id8: (map f id3)

Krok 11: Kroky 1 až 6 sú vykonané pre ident id8 s výsledkom, že dva zo stĺpcov bázovej matice sú uložené v pamäti:

id1: (2 3 id8)

id8: (id9 id10)

id9: (f -7)

id10: (f -8)

Krok 12: Sú vykonané kroky 7 až 9 pre ident id9 a id10 s výsledkom 8 a 9. Uloženie:

id1: (2 3 id8)

id8: (8 9)

čo znamená: (2 3 (8 9)) - funkcia f bola aplikovaná na všetky prvky v zozname argumentov.

Je nutné tiež poznamenať, že popísané kroky môžu byť vykonané jadrovými bunkami iným, oveľa účinnejším spôsobom. Napríklad namiesto ukladania medzivýsledkov v objektovom ukladaní môže byť vykonaná ďalšia redukcia/vykonanie v jadrovej bunke, kde je to vhodné.

Transpose

Formát: (transpose list)

Inštrukcia Transpose je načítaná do jedného z pomocných registrov, napríklad F0 a zoznam argumentov, napríklad zoznam zoznamov, je načítaný do matice báзовých registrov, vid' obr..18. Obsah matice báзовých registrov je transponovaný. Preto inštrukcia transpose, ktorá má trojúrovňovú štruktúru

(transpose,

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

.....

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

je vykonaná tak, že výsledok je zapísaný do dvojúrovňovej štruktúry

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

.....

$((e_{n,1}, \dots, e_{n,m}))$

Názorný príklad:

Štruktúra zoznamu

$((1\ 2\ 3\ 4),$

$(5\ 6\ 7\ 8),$

$(9\ 10\ 11\ 12),$

$(13\ 14\ 15\ 16))$, kde prvý zoznam, teda $(1\ 2\ 3\ 4)$, je uložený v prvom stĺpci bazových registrov, teda bazové registre $S_{0,0}$ až $S_{0,3}$, druhý zoznam, teda $(5\ 6\ 7\ 8)$, je uložený v druhom stĺpci bazových registrov, teda bazové registre $S_{1,0}$ až $S_{1,3}$, atď. je transponovaný na

$((1\ 5\ 9\ 13),$

$(2\ 6\ 10\ 14),$

$(3\ 7\ 11\ 15),$

$(4\ 8\ 12\ 16))$, kde prvý zoznam, teda $(1\ 5\ 9\ 13)$, je uložený v prvom stĺpci bazových registrov, teda v bazových registroch $S_{0,0}$ až $S_{0,3}$ atď.

Swap

Formát: (swap m list)

Inštrukcia swap (výmena) je vykonaná tak, že inštrukcia swap, ktorá má trojúrovňovú štruktúru

$(\text{swap } m ((e_{1,1}, \dots),$

$\dots$

$(e_{m,1}, \dots).$

$(e_{m+1,1}, \dots).$

$\dots$

$(e_{n,1}, \dots)))$

kde zoznam zoznamov s prvkami $e_{i,j}$, i a j sú označenia pozície prvku v matici bazového registra, je prepísaná na dvojúrovňovú štruktúru

$((e_{1,1}, \dots),$

$\dots$

$(e_{m+1,1}, \dots),$

$$(e_{m,1}, \dots),$$

$$\dots$$

$$(e_{n,1}, \dots))$$

tak, že prvok $(e_{m,1}, \dots)$ vymení miesto s prvkom $(e_{m+1,1}, \dots)$.

Skip

Formát: (skip m list)

Inštrukcia skip sa vykoná tak, že inštrukcia skip, ktorá má trojúrovňovú štruktúru

$$(\text{skip } m ((e_{1,1}, \dots),$$

$$\dots$$

$$(e_{m-1,1}, \dots)$$

$$\dots$$

$$(e_{m,1}, \dots)$$

$$(e_{m+1,1}, \dots)$$

$$\dots$$

$$(e_{n,1}, \dots)))$$

kde zoznam zoznamov s prvkami $e_{i,j}$, i a j sú označenia pozície prvku v matici báзовých registrov, je prepísaná na dvojúrovňovú štruktúru

$$((e_{1,1}, \dots),$$

$$\dots$$

$$(e_{m-1,1}, \dots),$$

$$(e_{m+1,1}, \dots),$$

$$\dots$$

$$(e_{n,1}, \dots))$$

tak, že zoznam $(e_{m,1}, \dots)$ je vynechaný.

4. Vyberanie zoznamu.

Cieľ, ktorý obsahuje zoznam, je umiestnený v hlavných registroch. Pokiaľ zoznam obsahuje prvky, ktoré sú vložené zoznamy, tieto zoznamy sú uložené vertikálne vo vedľajších registroch.

Operácia `expand_list` môže byť vykonaná v jednom cykle. Obsah báзовých registrov je posunutý diagonálne dole doprava o jeden

krok, s výnimkou obsahu v hlavnom registri, ktorý je prenesený na vertikálnu zbernicu a vložený do najhornejšieho vertikálneho registra v tomto stĺpci (obr. 19). Opakovaný expand_list môže byť použitý pre "naplnenie" hlavných registrov dátami.

Bitová bunka.

Celá pamäť, ktorá je časťou redukčného procesora podľa vynálezu, má byť implementovaná v technológii VLSI. Každá bitová bunka je uskutočnená tak, že ich návrh je prispôsobený VLSI technológii. Príklad obvodu bitovej bunky spoločne s budiacimi stupňami pre vodiče bitovej bunky d , d^* a acc sú ukázané na obr. 20. Bitová bunka 15 je na obrázku orámovaná čiarkovanou čiarou.

Každá bitová bunka je pripojená k riadiacej jednotke dvomi vodičmi, ako sú vodiče d a d^* v príklade bitovej bunky, ktorá je na obr. 20. Preto každá časť informačnej zbernice, ktorá ide na prvok pre uloženie informácie a ktorá je dlhá 38 bitov, obsahuje 76 vodičov. Každá časť takej zbernice je pripojená k poliam pre ukladanie 151, ktorá je umiestnená v stĺpci v oblasti pamätevej bunky.

Ako je vidieť z obr. 20, bitová bunka má iba štyri spoje a to prvý spoj V_{cc} , ktorým je zaistené trvalé napájanie, druhý, tretí a štvrtý spoj acc , d , d^* , z ktorých každý je nastaviteľný minimálne v štyroch rôznych riadiacich stavoch, ako bude podrobne popísané naskoršie.

Bitová bunka, ktorá je ukázaná na obr. 20, je CMOS bunka so štyrmi tranzistormi. Je statická a má záťaž. Záťaž je spoj emitor/kolektor tranzistora MOS FET $I1$ a $I2$, ktoré majú v tomto príklade invertované hradlá. Je tiež možné mať miesto toho odporovú záťaž (nie je ukázané).

Bunka je ovládaná klopným obvodom z každej strany. Medzi vodičom prístupu acc a napájacím vodičom V_{cc} sú paralelne dve sériové spojenia emitor/kolektor MOS FET a záťaž $T1, I1$ a $T2, I2$. Kolektor tranzistora $T1$ je pripojený k hradlu tranzistora $T2$ a kolektor tranzistora $T2$ je pripojený k hradlu tranzistora $T1$. Dióda $D1$ je zapojená medzi vodič d a prepojenie $n1$ medzi

kolektorom tranzistora T1, záťaže I2 a hradlom tranzistora T2. Dióda d1 je zapojená medzi vodič d* a prepojenie n2 medzi kolektorom tranzistora T2, záťažou I2 a hradlom tranzistora T1. Ku každej z diód D1 a D2 je pripojený tranzistor MOS FET. Tieto majú kolektor a hradlo navzájom prepojené a pripojené na vodič d alebo d*.

Podstatné kvality prvkov obvodu sú také, že diódy D1 a D2 sú prvky, ktoré umožňujú, aby prúd tiekol iba jedným smerom vzhľadom k vodičom d a d* a že tranzistory T1 a T2 sú aktívne prvky, pomocou ktorých môže byť prúd riadený zmenou potenciálu na ich hradlách. Prepojenia n1 a n2 sú uzly, na ktorých je potenciál, ktorý sa týka jednobitovej informácie. Každá záťaž I1 a I2 je prvok, ktorý sa chová ako odpor.

Na obr. 20 je uvedené napätie V_{CC} s vysokým potenciálom. Diódy D1 a D2 sú potom nasmerované tak, že prúd tečie od vodičov d alebo d* k uzlom n1 alebo n2. Odpor aktívnych prvkov T1 alebo T2 sa znižuje, keď potenciál na hradlovej elektróde sa zvyšuje a tak majú uzly nižší potenciál. V iných príkladoch však potenciály a prúdy môžu byť zvolené tak, aby mali opačný smer ako ten, ktorý je uvedený na príklade podľa obr. 20.

Bitová bunka si môže zapamätať hodnotu V_{store} , hodnotu, ktorá je alebo "áno" alebo "nie". Bitová bunka má takú štruktúru, že je nastaviteľná rôznymi funkčnými stavmi nastavením rôznych potenciálov na vodičoch acc, d a d*.

Riadiace stavy sú vysoká úroveň, nízka úroveň, prúd do bunky pre všetky vodiče a tiež prúd von z bunky pre vodič acc. Vodič acc je prístupový vodič, ktorý ide od hlavy 16 a je pripojený ku všetkým bitovým bunkám 15 v poli pamätevej bunky. Tretí a štvrtý vodič d a d* majú navzájom invertované signály, keď prebieha zápis alebo čítanie z bunky a prístupový vodič acc je LOW.

Riadiace a snímacie zosilovače v prvkovej hlave 16.

Riadiace a snímacie zosilovače v hlave 16 sú schématicky znázornené v čiarkovanom rámečku na obr. 20. Prvý tranzistor T3 má svoj emitor pripojený na napätie V_r , svoj kolektor k prístupo-

vému vodiču acc vo všetkých bitových bunkách 15 v poli pamäťových buniek 151 a na hradlo je privedený hodinový signál prech z hodinového signálového generátora (nie je uvedené). Druhý tranzistor T4 má svoj emitor pripojený na napätie 0V, kolektor k prístupovému vodiču acc vo všetkých bitových bunkách 15 v poli pamäťových buniek 151 a na hradlo je privedené napätie V3, ktoré je vysoké, keď napätie 0V bude nastavené na prístupovom vodiči acc. Ako už bolo skôr uvedené, vodič acc bude pripojený ku všetkým bitovým bunkám prvku spínania, teda napríklad 38 bitovým bunkám. Je to zaistené v hlave. Preto by hlava mala byť malá. Úroveň napätia vodiča acc je zosilnená v zosilovači AMP.

Riadiace a snímacie zosilovače v interfejsse 7.

Riadiace a snímacie zosilovače v interfejsse 7 pre vodiče bitovej bunky d a d* na obr. 20 sú schématicky znázornené v inom rámečku z čiarkovanej čiary. Je však nutné si všimnúť, že obvody iba znázorňujú jeden možný spôsob riadenia a snímania vodičov d a d*. Vstup/výstup IN/OUT je pripojený k jadrovej bunke 2, uvedený na obr. 1. Preto je obvod jeden z mnohých podobných obvodov, ktoré môžu byť použité v interfejsse 7 medzi objektovým ukladaním 1 a jadrovou bunkou 2.

Obvod pre zápis pre vodič d obsahuje prvý pár tranzistorov T5 a T6, prvý z nich je v príklade typu n a druhý typu p a majú svoje konektory zapojené k vodiču d a tvoria tak napäťový delič. Tranzistor T5 má svoj emitor pripojený na potenciál Vr a na svoje hradlo má privedený prednabíjací signál prech. Druhý tranzistor T6 má svoj emitor pripojený na potenciál Vcc a na svoje hradlo má privedený riadiaci signál V4, ktorý ide do nuly, keď potenciál Vcc bude privedený na vodič d. Obvod pre zápis pre vodič d tiež obsahuje sériovo zapojený emitor/kolektor tranzistora T9 typu p a tranzistor T10 typu n, ktoré sú zapojené medzi napäťový zdroj Vcc a kolektor tranzistora T11 typu n, ktorý má svoj emitor pripojený na zem a svoje hradlo má pripojené ku vstupu zápisu z externého riadenia. Prepojenie medzi kolektormi tranzistorov T9 a T10 je pripojené k hradlu tranzistora T6 a má

napätie V_4 . Na hradlo tranzistora T9 je privedený invertovaný prednabíjací signál $prech^*$, ktorý spojuje hradlo tranzistora T6 ku zdroju napätia V_{cc} cez vodivý tranzistor T9 v priebehu fáze prednabíjania.

Obvod pre zápis pre vodič d^* obsahuje druhý pár sériovo radených tranzistorov T7 a T8. V danom zapojení je prvý typu n a druhý typu p, ich kolektory sú zapojené k vodiču d^* a tiež tvoria napäťový delič. Tranzistor T7 má svoj emitor pripojený na potenciál V_r a na svoje hradlo má privedený prednabíjací signál $prech$. Druhý tranzistor T8 má svoj kolektor pripojený na potenciál V_{cc} a na svoje hradlo má privedený riadiaci signál V_5 , ktorý ide do nuly, keď potenciál V_{cc} bude privedený na vodič d^* .

Externý vodič IN/OUT pre vstup a výstup je pripojený k dvom trojstavovým invertorom. Jeden z trojstavových invertorov, ktorý má svoj výstup zapojený na vodič IN/OUT, obsahuje sériovo zapojený spoj emitor/kolektor dvoch tranzistorov T14 a T15 typu n a dvoch tranzistorov T16 a T17 typu p. Hradlo tranzistora T16 je pripojené na riadiaci vodič, ktorý zaisťuje signál $bitin$ a na hradlo tranzistora T15 je privedený invertovaný signál $bitin^*$. Druhý z trojstavových invertorov, ktorý má svoj vstup zapojený na vodič IN/OUT obsahuje sériovo zapojený spoj emitor/kolektor dvoch tranzistorov T18 a T19 typu n a dvoch tranzistorov T20 a T21 typu p. Hradlo tranzistora T19 je pripojené na riadiaci vodič, ktorý zaisťuje signál $bitin$ a na hradlo tranzistora T20 je privedený invertovaný signál $bitin^*$. Výstup druhého trojstavového invertora je pripojený k hradlu tranzistora T10.

Čítací zosilovač, ktorý obsahuje tranzistor T22 typu n, ktorý má svoj emitor pripojený na zem, na hradlo má privedené konštantné napätie V_{bias} , ktoré udržiava tranzistor T22 trvale otvorený a funguje ako prúdový generátor a svoj kolektor má pripojený k paralelnému zapojeniu dvoch sériovo zapojených spojov emitor/kolektor tranzistorov typu n a tranzistorov typu p T23, T24 a T25, T26, ktoré majú svoje druhé konce pripojené na zdrojové napätie V_{cc} . Hradlá tranzistorov T24 a T25 typu p sú prepojené a pripojené k prepojeniu kolektorov T23 a T24. Hradlo tranzistora

T23 je pripojené na vodič d a hradlo tranzistora T25 je pripojené na vodič d*.

Každú hodinovú periódu sú signály prech a prech* rozdelené na prednabíjaciú fázu, v ktorej má signál prech vysokú úroveň a fázu make, kedy má signál prech nízku úroveň a ostatné riadiace signály určujú operáciu, ktorá sa má vykonať. Preto sú vo fáze prednabíjania vodiče d, d* a acc dopredu nabité na napätie Vr cez tranzistory T5, T7 a T3.

Signály bitin a bitin* rozhodujú, kedy majú byť dáta vyslané z a do bitovej bunky 15. Keď má signál bitin nízku úroveň a signál bitin* vysokú úroveň, budú dáta prenesené z bitovej bunky na vodič IN/OUT cez prvý trojstavový invertor.

Keď má signál bitin vysokú úroveň a signál bitin* nízku úroveň, budú dáta prenesené do bitovej bunky z vodiča IN/OUT cez trojstavový invertor.

Pri operácii čítania vo fázi dve po prednabití vodičov d, d* a acc na Vr sú vodiče d a d* ponechané v treťom stave (plávajúce) a vodič acc je nastavený na napätie 0V pomocou vysokého napätia V3, ktoré udržiava tranzistor T4 v otvorenom stave. To spôsobí, že uzol bude mať najnižší potenciál, povedzme n1, ktorý má byť znížený na potenciál medzi Vr a 0V. Z toho dôvodu tečie prúd z vodiča d do uzla n1 do vodiča acc. Tento prúd vybíja vodič d, teda napätie na vodiči sa znižuje. Táto redukcia napätia je meraná čítacím zosilovačom T22 až T26. Výsledok čítania sa objaví na prepojení medzi kolektormi tranzistorov T25 a T26 a je privedené na vstup prvého trojstavového invertora T14 až T17. Signál bitin má nízku úroveň a bitin* má vysokú úroveň a zaisťujú prenos prečítanej a zosilnenej bitovej hodnoty na vstupný/výstupný vodič IN/OUT. Je dôležité, aby vodiče d a d* neboli riadené aktívnym spôsobom behom fázy dve, pretože by sa nezískala žiadna redukcia napätia na jednom z vodičov.

Preto pre operáciu čítania obidva vodiče d a d* sú na začiatku nastavené na potenciál Vr. Vodiče d a d* sú pevne držané na potenciáli Vr, ale jeden z nich trochu poklesne z dôvodu "prúdu do" bunky, ktorá vybíja jeden z vodičov d, d*. Pretože Vr je tu

definované ako "nízka úroveň", nízky potenciál bude nižší ako "low". Vodiče d a d^* dávajú hodnoty čítania. Ak je d nižšie ako d^* , dostane sa NIE, keď je d vyššie ako d^* , dostane sa hodnota ÁNO. Pre operácie don't write, write false, write true, don't write a don't comp. informačné potenciály na vodičoch d a d^* nedávajú žiadnu informáciu.

Pre operáciu zápisu vo fáze dve po prednabití vodičov d, d^* a acc na V_r je vodič acc nastavený na napätie $0V$ vysokým napätím V_3 , ktoré otvorí tranzistor T4. Hodnota, ktorá má byť uložená, je na vodiči vstupu/výstupu IN/OUT. Signály bitin v jednotke a bitin* v nule aktivujú druhý trojstavový invertor T18 až T21 na prenos hodnoty na vodiči IN/OUT na jeho výstup. Riadiaci signál zápisu, ktorý je v jednotke na hradle tranzistora T11 pripojí emitor tranzistora T10 a T13 na $0V$.

Signál vysokej úrovne z druhého trojstavového invertora T18 až T21, teda keď má byť zapísaná "0" alebo stav "nie", otvorí tranzistor T13, nastaví napätie V_5 na nízku úroveň, tranzistor T8 je riadený tak, že je otvorený a vodič d^* je nastavený na napätie V_{cc} , teda na vysokú úroveň. Invertovaný signál z druhého trojstavového invertora, privedený na hradlo tranzistora T10, ktorý má nízku úroveň, ho bude udržiavať v nevodivom stave, napätie V_4 , ktoré je pripojené na napäťový zdroj V_{cc} počas prednabíjacej fázy, bude udržiavané na tomto napätí. Tranzistor T6 bude udržiavaný v zavretom stave a napätie V_r , pripojené na vodič d počas intervalu prednabíjania cez tranzistor T5, bude udržiavané.

Signál nízkej úrovne z druhého trojstavového invertora T18 až T21, teda keď má byť zapísaná "1" alebo stav "áno", bude ovládať obvod T5, T6, T9, T10 pre vodič d tak, aby bol nastavený na napätie V_{cc} vysokej úrovne cez invertor INV, zatiaľ čo obvod pre zápis T7, T8, T12, T13 bude udržiavať vodič d^* na napätí V_r , na ktoré bol nastavený v priebehu fázy prednabíjania.

Ako je zrejmé z uvedených príkladov, uzly uloženia n_1 a n_2 sú v zapojení na obr. 20 použité v nasledujúcom spôsobe operácie. Jeden alebo obidva uzly n_1, n_2 sú nabité alebo vybité počas druhej fázy operačného cyklu v závislosti na tom, ktorý z

riadiacich signálov V3, V4 a V5 má byť použitý, teda či vodič acc je nastavený na 0V alebo či jeden (alebo obidva) vodiče d a d* sú nastavené na Vcc.

Ako už bolo uvedené, každý operačný cyklus je zložený z periódy prednabíjania a periódy vykonania. Preto keď je ďalej uvedené, že vodič acc je nastavený na vysokú úroveň znamená to, že signál V3 neovláda tranzistor T4 tak, aby počas periódy vykonávania na vodiči acc bolo nastavené napätie 0V. Podobne keď je uvedené ďalej, že riadiaci signál V4 alebo V5 neovláda tranzistor T6 alebo T8 tak, aby bol vo väzobnom stave cez napätie Vcc, ktoré je väčšie ako napätie Vr, na vodič d alebo d* počas periódy vykonávania. Ale keď je vodič d alebo d* nastavený na vysoký stav, potom tranzistor T6 alebo T8 bude riadený tak, aby vodič pripojil na napätie Vcc.

Oblasť pamätevej bunky by mohla byť pomerne rozsiahla, napríklad môže obsahovať 256 pamäťových buniek, čo znamená, že každý pár tranzistorov T5, T6 a T7, T8 je pripojený na vodič, ktorý obsluhuje jednobitovú bunku vo všetkých pamäťových bunkách, napríklad 256 bitových buniek. Preto počet tranzistorov musí byť nastavený podľa celkovej kapacity zbernice a požadovanej rýchlosti.

Napätie Vr môže byť získané zo skratovaného invertora, aby sa udržal známy vzťah Vr a snímacieho zosilovača invertora. Prístupové obvody v hlave budú riadiť bitové bunky a tiež snímať informáciu z bitových buniek.

Nasledujúce funkčné stavy je možné nastaviť riadiacimi stavmi:

rest:	bunka si pamätá hodnotu vstore
read false:	hodnota vstore = je možné čítať "nie"
read tru:	hodnota vstore = je možné čítať "áno"
don't read:	bunka si pamätá hodnotu vstore
write fals:	uložená hodnota vstore je nastavená na "nie"
write true:	uložená hodnota vstore je nastavená na "áno"
don't write:	bunka si pamätá hodnotu vstore
comp. false:	uložená hodnota je porovnaná s hodnotou "nie"

comp. true: uložená hodnota je porovnaná s hodnotou "áno"

don't comp.: bunka má iba uloženú hodnotu vstore

Ďalej je uvedená tabuľka operácií pre rôzne režimy operácií bitovej bunky:

Op. režim	acc	d	d*
rest	nízka	nízka	nízka
read false	nízka	prúd dovnútra	vysoká
read true	nízka	vysoká	prúd dovnútra
don't read	vysoká	ľubovoľná	ľubovoľná
write false	nízka	nízka	vysoká
write true	nízka	vysoká	nízka
don't write	vysoká	ľubovoľná	ľubovoľná
comp. false	ľubovoľná	nízka	vysoká
comp. true	ľubovoľná	vysoká	nízka
don't comp.	ľubovoľná	nízka	nízka

Pre comp. false a comp. true vodič acc by mal mať stavový prúd von, pokiaľ výsledok komparácie je DIFFERENT.

Pre operáciu comp. false alebo comp.true vodič acc (prístupový vodič) dáva výsledok komparácie. Vodič acc je dopredu nabitý na Vr a vstupné dáta sú privedené na vodič d a ich negácia na vodič d*. Pokiaľ hodnota, uložená v bitovej bunke je iná ako vstupné dáta, vodič acc bude nabitý cez jednu z diód D1 alebo D2 a cez odpovedajúci tranzistor T1 alebo T2 typu n. To je zaistené zosilovačom AMP v hlave 16. Keď je pri komparácii zistené FIT, vodič acc bude držaný na potenciáli Vr.

Výrazy "prúd dovnútra" a "prúd von" vyjadrujú, že náboj sa v danom okamžiku premiestňuje dovnútra respektíve von z uvedeného vodiča. To je v režime operácie REST obyčajne vykonané inicializáciou vodiča do stavu HIGH alebo LOW a potom sa urobí zmena do aktuálneho režimu. Prúd potom nabije alebo vybiže príslušný vodič. Keď neexistuje žiadny prúd, neprenesie sa žiadna výrazná zmena. Preto behom časovej postupnosti nenastane žiadna napäťová zmena.

Obvod riadenia objektového ukladania.

Dekodér priority (obr.21A a B)

Dekodér priority 11,ktorý je použitý na obr.21A a 21B,je rozdelený na štvorbloky.Ako je uvedené na obr.21A,každý blok má jeden pár ľavých vodičov granta a reqa a štyri páry pravých vodičov reg0, grant0, ...až req3, grant3.

Ako je uvedené na obr.21B,prvý štvorblok 520 má svoje štyri páry pravých vodičov zapojené ku každému páru ľavých vodičov štyroch štvorblokov, z ktorých sú uvedené vonkajšie 521 a 522.Bloky 521,522 a bloky 520,522 sú spojené s invertujúcimi zosilovačmi 523 a 524,zosilovač 523 dáva informáciu do nižšieho bloku v reťazci blokov,aká je požiadavka priority potrebná pre blok vyššie v reťazci a zosilovač 524 dáva informáciu do bloku vyššie v reťazci blokov,ktorému je vyhovené. V druhom štvorblokovom stĺpci sú teda štyri bloky 521,.....522.

Každý blok v druhom stĺpci je pripojený ku štyrom štvorblokom v treťom stĺpci blokov rovnakým spôsobom,ako bloky v druhom stĺpci sú pripojené ku štvorblokom 520.Počet štvorblokov v treťom stĺpci bude teda šestnásť.Ukázané sú iba štvorbloky 525 a 526 ktoré sú najviac na okraji.

Každý blok v treťom stĺpci je potom pripojený ku štyrom štvorblokom vo štvrtom stĺpci blokov rovnakým spôsobom.Počet štvorblokov v treťom stĺpci potom bude 64.Uvedené sú štvorbloky 527 a 528,ktoré sú najviac na okraji.

Pravé vodiče štvorblokov v štvrtom stĺpci sú pripojené k objektovému ukladaniu.Každý pár je prispôsobený k tomu,aby slúžil ako zbernica 18 a 19 pre pamäťovú bunku 10,ako je zrejmé z obr.21B.

85 štvorblokov je vytvorené pre obsluhu 256 spínačov.Najnižší blok 528 obsluhuje najnižšie pamäťové bunky až k pamäťovej bunke 0 a najvyšší blok 527 obsluhuje najvyššie,teda až pamäťovú bunku 255.

Konfigurácia,ktorá je uvedená na obr.21A a 21B,používa dominovú prednabíjanú logiku,kde plný dekodér priority obsahuje

kaskádované dominové stupne, ktoré odpovedajú prenosu signálu s požiadavkou napríklad od reg0 v najnižšom bloku 528 cez všetky bloky dekodérov priority a späť, aby dali chybný signál grant pre všetky pamäťové bunky, okrem pamäťovej bunky číslo 0.

Ako je uvedené na obr. 21A, každý blok obsahuje 5 radov tranzistorov MOS FET a každý rad obsahuje o jeden MOS FET viac ako rad nižšie, okrem piateho radu, ktorý má rovnaký počet tranzistorov MOS FET ako štvrtý rad.

Každý z tranzistorov MOS FET $Tr_{0,0}$ až $Tr_{3,0}$ najviac vpravo v štyroch najnižších radoch je uvedený ako typ p a má svoje hradlo pripojené ku zdroju hodinového signálu, kolektor pripojený ku kladnému napájaciemu napätiu a emitor pripojený ku grant vodiču grant0, grant1, grant2 a grant3. MOS FET tranzistor $Tr_{4,0}$ v najvyššom rade má svoj emitor namiesto vodiča granti, kde i je číslo medzi 0 až 3, pripojený k vodiču reqa, ktorý je pripojený k najnižšiemu bloku v kaskáde blokov cez invertor 523.

Zvyšok tranzistorov MOS FET $Tr_{0,1}$, ktoré sú typu n v prvom rade, $Tr_{1,1}$ a $Tr_{1,2}$ v druhom rade, $Tr_{2,1}$, $Tr_{2,2}$, $Tr_{2,3}$ v treťom rade, $Tr_{3,1}$, $Tr_{3,2}$, $Tr_{3,3}$, a $Tr_{3,4}$ vo štvrtom rade, $Tr_{4,2}$, $Tr_{4,3}$, $Tr_{4,4}$ a $Tr_{4,5}$ v piatom rade majú svoje emitory pripojené na zem a svoje kolektory pripojené ku grant vodiču grant1, grant2 a grant3 pre štyri najnižšie rady a k vodiču reqa pre piaty rad.

Vodič granta je pripojený k hradlu každého tranzistora $Tr_{0,1}$, $Tr_{1,1}$, $Tr_{2,1}$, $Tr_{3,1}$. Vodič req0 je pripojený k hradlu každého MOS FET tranzistora $Tr_{1,2}$, $Tr_{2,2}$, $Tr_{3,3}$ a $Tr_{4,2}$. Vodič req1 je pripojený k hradlu každého MOS FET tranzistora $Tr_{2,3}$, $Tr_{3,3}$ a $Tr_{4,3}$. Vodič req2 je pripojený k hradlu každého MOS FET tranzistora $Tr_{3,4}$ a $Tr_{4,4}$. Vodič req3 je pripojený k hradlu MOS FET tranzistora $Tr_{4,5}$.

Dekodér priority pracuje v dvoch fázach. V prvej fáze, kedy je hodinový signál v nule, všetky vodiče granti sú prednabité na vysokú úroveň (áno). Potom majú všetky signály reqi nízku úroveň (nie sú potrebné). V druhej fáze je prednabíjanie uzavreté, teda hodinový signál je v jednotke. Potom niektorý ľubovoľný výstup regi prejde do vysokej úrovne, čo nastaví všetky granti vyššie do

nízkej úrovne (nie je zvolené) a nastaví reqa na nízku úroveň. Keď reqa prejde na nízku úroveň, reqi v susednom štvorbloku naľavo na obr.21 je nastavené na vysokú úroveň. Signály reqa a granta v štvorbloku 520 nemajú žiadny význam. Ale grant v štvorbloku 520 je pripojený na zem. Signál reqa v štvorbloku 520 dáva výsledok "ANY" na vodiči 14 na obr.2 a 3, pretože musia prejsť na nízku úroveň, keď niektorý reqi, pripojený k ľubovoľnému bunkovému spínaču prejde do vysokej úrovne.

Prvková hlava.

Podrobné zapojenie prvkovej hlavy 16 je uvedené na obr.22. Prvková hlava 16 riadi prístupový vodič acc k pamäťovým bunkám 10, sníma prístupový vodič acc a vodič ANY 14, tu nazývaný any_type, vykonáva operácie wired_and a wired_nor na zberniciach a a b a tiež číta zbernice a a b. Okrem toho obsahuje interný dynamický pamäťový bit.

MOS FET tranzistor n0 s kanálom n má svoj emitor/kolektor pripojený medzi napätie Vr, ktoré je rovnaké ako napätie Vr na obr.20 a prístupový vodič acc. Hodinový pulz cpb je privedný k hradlu MOS FET tranzistora n0. MOS FET tranzistor n1 s kanálom n má svoj emitor/kolektor pripojený medzi zem a prístupový vodič acc.

Sériové zapojenie paralelne spojených emitorov/kolektorov dvoch MOS FET tranzistorov p2 a p3 s kanálom n, emitor/kolektor MOS FET tranzistora p4 s kanálom p a emitor/kolektor MOS FET tranzistora n5 s kanálom n je zapojený medzi napätie zdroja Vcc a zem.

Vodič any_type, priamo zapojený k najbližšej bitovej bunke, je pripojený ku hradlu MOS FET tranzistora p2. Drôtový pár z centrálnej riadiacej jednotky je pripojený ku hradlu MOS FET tranzistora p3. Prístupový drôt acc je zapojený k hradlu MOS FET tranzistora p4 cez invertor INV1 vrátane sériovo zapojeného emitora/kolektora MOS FET tranzistora p1 s kanálom p a n6 s kanálom n, ktoré majú svoje hradlá zapojené k prístupovému drôtu acc. MOS FET tranzistor n7 s kanálom n má svoj emitor/kolektor zapojený medzi invertor

INV1 a zem.Drôt eval.s,teda evaluation select (voľba vyhodnotenia) z centrálnej riadiacej jednotky je pripojený ku hradlu MOS FET tranzistora n7.Vodič set.s,teda set select (nastavenie voľby),z centrálnej riadiacej jednotky je pripojený ku hradlu MOS FET tranzistora n5.

Prepojovací bod i1,nazývaný tiež uzol voľby,je medzi medzi kolektormi MOS FET tranzistorov p4 a p5 spojený s kolektorom MOS FET tranzistora p6 s kanálom p,ktorý má svoj kolektor pripojený cez emitor/kolektor MOS FET tranzistora p7 s kanálom p na napätie zdroja +Vcc.Vodič reset.b z centrálnej riadiacej jednotky je pripojený na hradlo MOS FET tranzistora p6.Vodič b,ktorý je rovnaký ako vodič b na obr.3,je pripojený na hradlo MOS FET tranzistora p7.

MOS FET tranzistor p8 s kanálom p má svoj kolektor pripojený na vodič a,ktorý je rovnaký ako vodič a (uvedené na obr.3) a svoj emitor má pripojený na prepojovací bod i2.Vodič Wand.a,teda wired_and_a,z centrálnej riadiacej jednotky je pripojený na svoje invertované hradlo.MOS FET tranzistor p9 s kanálom p má svoj kolektor pripojený na vodič b a svoj emitor na prepojovací bod i2.Vodič Wand.b,teda wired_and_b z centrálnej riadiacej jednotky je pripojený cez svoje invertované hradlo.MOS FET tranzistor p10 s kanálom p má svoj emitor pripojený cez zdroj napätia +Vcc.Prepojovací bod i1 je cez invertor INV2 pripojený na invertované hradlo MOS FET tranzistor p10.Invertor INV2 obsahuje sériovo zapojený emitor/kolektor MOS FET tranzistora p14 s kanálom p a MOS FET tranzistora n8 s kanálom n medzi napätie zdroj napätia +Vcc a zem a majú svoje hradlá pripojené na prepojovací bod i1.

Sériové zapojenie spoja emitor/kolektor MOS FET tranzistorov p11 a p12 s kanálom p je pripojené medzi vodič a a napäťový zdroj +Vcc.Vodič Vor,teda wired_or z centrálnej riadiacej jednotky je zapojený na hradlo MOS FET tranzistora p12.Prepojovací bod i1 je zapojený na hradlo MOS FET tranzistora p11.MOS FET tranzistor p13 má svoj spoj emitor/kolektor pripojený medzi vodič a a prepojovací bod i1.Vodič s.a,teda voľba a, z centrálnej riadiacej

jednotky je zapojený na hradlo MOS FET tranzistora p13.

Sériové zapojenie spoja emitor/kolektor MOS FET tranzistora n2 s kanálom n, MOS FET tranzistor p15 s kanálom p, MOS FET tranzistor p16 s kanálom p, je zapojený medzi zem a napäťový zdroj +Vcc. Prepojovací bod i3 medzi emitormi MOS FET tranzistorov n2 a p15 je pripojený na hradlo MOS FET tranzistora n1. Sériové zapojenie dvoch MOS FET tranzistorov p17 a p18 s kanálom p je peipojené na prepojovací bod i3 a napäťový zdroj +Vcc. Prepojovací bod i1 je pripojený na hradlo MOS FET tranzistora 18. Vodič r/w.b, teda read/write b, z centrálnej riadiacej jednotky je zapojený na invertované hradlo MOS FET tranzistora p15. Vodič r/w.s, teda read/write select, z centrálnej riadiacej jednotky je pripojený na hradlo MOS FET tranzistora p17. Vodič r/w.r, teda read/write reset, ktorý je použitý na resetovanie uzla i3 po operácii čítania alebo zápisu, z centrálnej riadiacej jednotky je pripojené na hradlo MOS FET tranzistora n2.

Funkcia logiky v zapojení prvkovej hlavy na obr.22 je nasledujúca. MOS FET tranzistor n0 prednabíja prístupový vodič acc pri každom negatívnom hodinovom impulze a MOS FET tranzistor n1 vyhodnocuje nízku úroveň pri čítaní alebo zápise. MOS FET tranzistor n2 prednabíja i3 na nízku úroveň, aby udržal MOS FET tranzistor n1 zatvorený, teda v nevodivom stave, pri pohotovostnom stave.

MOS FET tranzistory p17 a p18 vykonávajú čítanie/zápis riadený voľbou uzla. To sa používa napríklad v inštrukciách typu match, ktoré nulujú značku spínacej bunky alebo čítanie identity spínacej bunky a pokiaľ sa nenastavuje značka, tam kde sa vykonávajú rôzne činnosti v závislosti na informáciách na zberniaciach a a b.

Funkcia match porovnáva hodnotu napríklad takzvanú cieľovú hodnotu (goal) v jadrovej bunke a v pamäťovej bunke a porovnáva tieto hodnoty pri definovaní dvoch súborov. Výsledok porovnania je typu "nie", keď sa súbory nepretínajú. Tento prípad sa tiež predpokladá, keď časti spínača nie sú celkom definované.

MOS FET tranzistor n5 prednabíja uzol i1 a MOS FET

tranzistory p2 až p4 ho vyhodnocujú pri riadení z riadiacej jednotky 6 na hradle MOS FET tranzistora n7. MOS FET tranzistory p8 až p10 vykonávajú wired_and na zbernici a a/alebo na zbernici b v závislosti na riadiacich signáloch z centrálnej riadiacej jednotky. MOS FET tranzistory p11 a p12 vykonávajú wired_or na zbernici a pri riadení z centrálnej riadiacej jednotky.

Invertor INV2 invertuje bit voľby v uzle voľby. To je nutné, pretože sa vykonáva ako wired_or tak i wired_and. MOS FET tranzistor p13 vykonáva prenos hodnoty na zbernici a do uzla voľby. Iba stav vysokej úrovne sa prenáša, pretože ten bol prednabitý na nízku úroveň. Nakoniec MOS FET tranzistory p6 a p7 sa používajú na voľbu vysokej úrovne pri riadení. To je nutné pre inštrukciu čítania označenej pamäťovej bunky a pre vynulovanie značky tejto bunky, pretože potom bude uzol voľby resetovaný súčasne s vykonaným čítaním. Táto funkcia môže byť tiež použitá pre ostatné typy inštrukcií, v ktorých sa musí vykonávať logická operácia AND v prvkovej hlave.

Signál any_type je priamo pripojený na najbližšiu bitovú bunku. Obsahuje typ uloženej hodnoty. Počas porovnávania je MOS FET tranzistor p3 držaný zavretý, takže hodnota vysokej úrovne na vodiči, ktorý má signál any_type, bude generovať pravdivý súhlas. Podobne pre testovanie, ak je pamäťová bunka nepoužitá, vysoká úroveň na vodiči, ktorý má signál any_type a keď MOS FET tranzistor p3 je držaný zavretý, bude mať za následok, že signál voľby v uzle voľby zostane na nízkej úrovni. Pre test zhody je MOS FET tranzistor p3 držaný otvorený, teda je vo vodivom stave.

Signál any_type je priamo pripojený na najbližšiu bitovú bunku. Obsahuje typ uloženej hodnoty. Počas porovnávania je MOS FET tranzistor p3 držaný zatvorený, takže hodnota vysokej úrovne na vodiči, ktorý má signál any_type, bude generovať pravdivý súhlas. Podobne pre testovanie, ak je pamäťová bunka nepoužitá, vysoká úroveň na vodiči, ktorý má signál any_type a keď MOS FET tranzistor p3 je držaný zatvorený, bude mať za následok, že signál voľby na uzle voľby zostane na nízkej úrovni. Pre test zhody je MOS FET tranzistor p3 držaný otvorený, teda je vo vodivom stave.

Hlava spínača (obr.23).

Zapojenie hlavy spínača 17 na obr.23 je schopné vykonávať operáciu priority a operáciu MODE na zbernici a alebo b. Obidve operácie môžu byť vykonávané súčasne každá na inej zbernici. Operácia MODE, teda operácia na zbernici 12 (MODE) v obvode podľa obr.2 a 3, je citlivá buď na nízku alebo vysokú úroveň zbernice a, zatiaľ čo na zbernici b je citlivá iba na nízku úroveň. Hlava spínača môže tiež resetovať globálny signál MODE na zbernici 13 (obr.2) a čítať a zapisovať späť staré dáta zbernice na zbernici a alebo b.

Ako obvod, ktorý má vykonávať operáciu MODE, tak i obvod, ktorý má vykonávať operáciu priority, sú navrhnuté podľa princípu dvojstupňového domina. Prvý stupeň určuje, ktorá zbernica má byť ako vstup a správnu polaritu a druhý stupeň vykonáva aktuálnu operáciu. To znamená, že hlava spínača pracuje v dvoch fázach, kde prvá fáza je prednabitie a druhá fáza je fáza operačná. Preto sú tu MOS FET tranzistory n20, n21, n22 a n23 s kanálom n, ktoré majú svoje hradlá riadené hodinovým signálom prech z centrálnej riadiacej jednotky. Tranzistor n20 prednabíja zbernicu a, n21 zbernicu b, n22 zbernicu reg pre dekodér priority a n23 kontakt net F1 do stavu "low".

Kontakťová sieť F1 sa skladá z dvoch paralelných vetví sériovo zapojených párov cesty emitor/kolektor MOS FET tranzistorov p20, p21 a p22, p23 s kanálom p. Kontakťová sieť F1 je na jednej strane zapojená na zem cez spoj emitor/kolektor MOS FET tranzistora n23 a na druhej strane na napäťový zdroj +Vcc. Zbernica a je zapojená na hradlo MOS FET tranzistora p20 a cez invertor INV20 na hradlo MOS FET tranzistora p23, na ktorú je tak privedený signál a*, čo je invertovaný signál zbernice a. Signál mode.a z centrálnej riadiacej jednotky je privedený na hradlo MOS FET tranzistora p21 a jeho invertovaný signál mode.a* z centrálnej riadiacej jednotky je privedený na hradlo MOS FET tranzistora p22. Kontakťová sieť F1 realizuje funkciu $a * mode.a$ versus $a * mode.a *$.

Sériové zapojenie spoja emitor/kolektor dvoch MOS FET tran-

zistorov p24 a p25 sú zapojené medzi napäťový zdroj +Vcc a zbernicu b. Zbernica a je pripojená na hradlo MOS FET tranzistora p24 a signál ba z centrálnej riadiacej jednotky je privedený na hradlo MOS FET tranzistora p25.

Uzol medzi sieťou F1 a kolektorom MOS FET tranzistora n23 je pripojený na hradlo MOS FET tranzistora n24 s kanálom n, ktorý má svoj emitor pripojený na zem a svoj kolektor na zbernicu 12 (MODE) ako je uvedené na obr. 2 a 3, kde je vytváraný mode-signal. Sériové zapojenie spoja emitor/kolektor dvoch MOS FET tranzistorov p26 a p27 s kanálom p je zapojené medzi napäťový zdroj +Vcc a zbernicu b. Mode-signal na zbernici 12 je cez dva invertory INV21 a INV22 a zosilovač mode-signal tiež pripojený na hradlo MOS FET tranzistora p27, signál mode.b je privedený z centrálnej riadiacej jednotky na hradlo MOS FET tranzistora p26.

Signal mode.a vytvára operáciu mode na zbernici a. Signal mode.a* vytvára tiež operáciu mode na zbernici a, ale tvorí na zbernici inverznú hodnotu a*. Signal ba nastavuje zbernicu do stavu "high", pokiaľ zbernica bola nastavená "low". Signal mode.b nastavuje zbernicu b do "high", pokiaľ mode zbernice bol "low". Signal prech sa používa na prednabíjanie zbernice a a b do stavu "low".

Preto sa voľba vykonania operácie MODE podľa signálu a alebo a* vykoná pomocou kontaktu net F1. Zbernica MODE 12 môže byť potom nastavená na zem priamo cez všetky hlavy spínača 17 v objektovom ukladaní. Riadiaci signal mode.b na hradle tranzistora p26 potom privedie zbernicu b do stavu vysokej úrovne. Týmto spôsobom je možné vytvoriť dva rôzne testy na zbernici - ako na zbernici a tak i na zbernici b. Keď výsledok testu na zbernici a pri aktuálnej operácii je "high", potom zbernica b môže byť nastavená na "high". Napríklad WIRED OR môže byť vytvorené na zbernici a alebo b a WIRED AND na druhej a logická operácia môže byť realizovaná podľa toho, či výsledok je napríklad "false", zbernica b bude tiež "false". Iná podmienka je vytvorenie inej operácie, založenej na výsledku na zbernici b.

Sériové zapojenie spoja emitor/kolektor dvoch MOS FET tran-

zistorov p28 a p29 s kanálom p je zapojené medzi napäťový zdroj +Vcc a zbernicu b. Signál grant z dekodéra priority 11 je cez inverter INV23 pripojený na jeden vstup NAND hradla NAND1 a signál req na druhý vstup. Hradlo NAND obsahuje dva MOS FET tranzistory n25 a n26 s kanálom n, ktoré majú svoje emitor/kolektor spoje zapojené sériovo a dva MOS FET tranzistory p32 a p34 s kanálom p majú svoje emitor/kolektor spoje zapojené paralelne a pripojené medzi kolektor MOS FET tranzistora n26 a napäťový zdroj +Vcc. Emitor MOS FET tranzistora n25 je pripojený na zem. Signál grant* je privedený na hradlo MOS FET tranzistorov n25 a p32. Výstup hradla NAND je pripojený na hradlo MOS FET tranzistora p28 a na hradlo MOS FET tranzistora p33 s kanálom p, ktorý má svoj emitor/kolektor spoj pripojený medzi napäťový zdroj Vcc a vodič MORE 13. Signál grant.b z centrálnej riadiacej jednotky je privedený na hradlo MOS FET tranzistora p29.

Sériové zapojenie spoja emitor/kolektor dvoch MOS FET tranzistorov p30 a p31 je zapojené medzi napäťový zdroj +Vcc a zbernicu req na dekodér priority 11. Signál prio z centrálnej riadiacej jednotky je privedený na hradlo MOS FET tranzistora p30 a zbernica b je pripojená na hradlo MOS FET tranzistora p31.

Signál prio z centrálnej riadiacej jednotky je použitý na vysielanie signálu request(požiadavka) do dekodéra priority 11, teda odovzdať signál req, ak je zbernica b v stave nízkej úrovne ("low"). NAND hradlo p32, p34, n25, n26 zisťuje, či signál req má vysokú úroveň a signál grant má nízku úroveň, teda či sa vyžaduje priorita hoci nie je zvolená. V takom prípade hradlo NAND prejde do stavu nízkej úrovne a nastaví zbernicu 13 (MORE) na "high", to znamená, že indikuje, že priorita nie nutná. Signál grant.b sa používa na to, aby sa zbernica b nastavila do stavu "high", keď signál req je "high" a signál grant je "low". Po tomto príde z centrálnej riadiacej jednotky signál prio a potom signál grant.b na jednej pamäťovej bunke v objektovom ukladaní bude mať zbernicu b v stave "low".

Číslicová ALU a reprezentácia hodnoty.

Úlohou vynálezu je uviesť procesor prvého rádu s konvenčným typom číslicovej ALU, ktorá by bola riadená riadiacou jednotkou 6 a ktorá by bola pripojiteľná na jadrovú bunku. Zapojenie číslicovej ALU 5, ktorá je zvlášť vhodná pre procesor prvého rádu podľa vynálezu, bude popísaná ďalej. Číslicová ALU uvedená ďalej:

- * je schopná redukovať číslicové výrazy bez použitia prechodného uloženia
- * je schopná pracovať s hodnotou s plávajúcou rádovou čiarkou ako s operandami, ktoré majú premennú zložku medzi mantisou a exponentom a táto zložka je kódovaná v kódovom poli
- * používa na zberanie aritmetické inštrukcie dané pre jej operandy v kódovom poli s premennou dĺžkou kódu
- * má kovarianciu medzi reprezentáciou v plávajúcej rádovej čiarky a typom integer
- * má úplnú kódovú reprezentáciu.

Číslicová ALU môže vykonávať aritmetické, logické a relačné operácie na prvkoch s číselnou hodnotou nasledujúcim spôsobom:

- a) vstupný zoznam je odovzdaný vrátane prvkov s číselnou hodnotou, ktoré majú byť spracované a informácie o inštrukcii
- b) operácia je vykonaná na prvkoch s číselnou hodnotou vo vstupnom zozname s použitím iba informácie o inštrukcii
- c) kalkulácia je vykonaná prepísaním prvkov vo vstupnom zozname
- d) výsledok je prezentovaný vo výstupnom zozname.

Najvhodnejší maximálny počet prvkov vo vstupnom a výstupnom zozname je štyri, čo odpovedá počtu hlavných registrov v jadrovej bunke, ktorá odovzdáva vstupný zoznam a prijíma výstupný zoznam. Jeden prvok zoznamu potom môže byť odovzdaný s reprezentáciou inštrukčného slova, dva prvky zoznamu môžu byť odovzdané ktoroukoľvek z dvoch reprezentácií číslicovej hodnoty a jedna môže byť použitá na odovzdanie medzivýsledku výpočtu v priebehu výpočtu, ktorý sa vykonáva niekoľkonásobným prepísaním vstupných prvkov.

Zoznam s výhodou obsahuje aplikáciu funkcie, v ktorej jeden z jej prvkov je inštrukčný kód a zvyšok sú argumenty pre inštrukciu. Inštrukcia je vykonaná prepísaním a recykláciou inšt-

rukčného kódu z výsledkového výstupného zoznamu do vstupného, až do dosiahnutia výsledku. Pri každom prepísaní môže byť zoznam prepísaný tak, aby obsahoval revidované slovo inštrukčného kódu, ak prichádza do úvahy pre aktuálny výpočet, po ktorom nasledujú slová hodnoty.

Je vhodné mať slová číselných hodnôt v kódovanom tvare pred skutočným spracovaním. Slová, ktoré predstavujú číselné hodnoty, sú najlepšie kódované tak, že predstavujú úplnú reprezentáciu číselnej hodnoty, teda každá kódovaná reprezentácia hodnoty odpovedá iba jednej interpretovanej hodnote. Prvé kódovanie môže byť odovzdané v tvare binárneho slova v hodnote typu integer. Druhé kódovanie môže byť odovzdané ako binárne slovo, ktoré reprezentuje hodnotu s plávajúcou rádovou čiarkou. Kódovanie je najlepšie také, že po kódovaní hodnoty s plávajúcou rádovou čiarkou sú reprezentácie odovzdané rovnakým spôsobom ako reprezentácia hodnoty typu integer. Týmto spôsobom žiadna z reprezentácií nemusí mať binárny tvar. Hardware môže byť taký, aby vykonal výpočet na transformovaných bitových vzorcoch implicitne podľa daných pravidiel, teda bez znalostí, ako bola reprezentácia vytvorená.

Lahký spôsob, ako vykonať reprezentáciu plávajúcej rádovej čiarky, ktorá má premennú šírku exponentu, je mať reprezentáciu binárnej kódovanej hodnoty v plávajúcej rádovej čiarky, ktorá má kódové pole, pole exponenta a pole mantisy, pričom kódové pole má údaj o polohe rozdelenia medzi poľom exponenta a poľom mantisy také, že polia exponentu a mantisy majú premennú dĺžku. Úplná reprezentácia môže byť daná použitím virtuálnej "1", teda "1" nie je fyzicky daná v reprezentácii v prednej časti slova pre časť mantisy s reprezentáciou v plávajúcej rádovej čiarky.

Hardwarová štruktúra číslicovej ALU.

Číslicová aritmeticko-logická jednotka ALU, ktorá je schématicky znázornená na obr. 24A, obsahuje centrálnu riadiacu jednotku 1a a pole aritmetickej jednotky 2A, ktoré vykonávajú aritmetiku na konkrétnych vstupoch/výstupoch NU_0 , NU_1 , NU_2 a NU_3 , kde každý je zbernica, ktorá má veľa (M) prenosových liniek, ktoré prenášajú hod-

notu bitu M , kde M je napríklad 32. Číslicová ALU je riadená riadiacou jednotkou 6. Číslicová ALU komunikuje s riadiacou jednotkou 6 pomocou nasledujúcich signálov: hodinový signál C_{ALU} , výstup bude A_{BUSY} , riadiaci vstup A_{CNTL} , stavový výstup A^{STATE} a výstup dekódovanej inštrukcie A_{INS} .

Interfejs číslicová ALU - jadrová bunka.

Zbernica in/output NU_0 , NU_1 , NU_2 a NU_3 sú pripojené na hlavné alebo cieľové registre v jadrovej bunke v procesore prvého rádu podľa obr.1, v ktorom sú spracovávané inštrukcie redukčného typu.

Hlavná registrová bunka v jadrovej bunke, teda $S_{0,0}$, $S_{1,0}$, $S_{2,0}$, $S_{3,0}$ na obr.9 majú všetky NU svorku (obr.10), ktorá je pripojená na jednu zbernicu in/output NU_0 až NU_3 . Obsah hlavného registra $S_{x,0}$ tak môže byť prenesený na číslicovú ALU s použitím in/output zbernice NU_x , čo je spôsob, ako je vstupný zoznam odovzdaný číslicovej ALU, kde x je číslo medzi 0 a 3. Ďalej číslicová ALU môže odovzdať výsledok ako výstupný zoznam do hlavného registra, ktorý je umiestnený v rovinách NUM, teda slovo num je prenesené do číslicovej ALU.

Na in/output zberniciach NU_0 až NU_3 prenášajú zoznam slov num, čo môže byť číselná hodnota alebo inštrukcia. Všetky zbernice nemusia byť vždy používané. Vstupný zoznam na in/output NU_0 až NU_3 obsahuje informáciu o druhu výpočtu, ktorý má byť vykonaný a riadiaca jednotka 1a odvodzuje túto informáciu od zoznamu počas riadenia rôznych zložiek v poli 2a. Výpočet je vykonávaný prepísaním obsahu vo vstupnom zozname po bitoch alebo skupinách bitov, teda keď bity majú rovnakú polohu v slove samostatne alebo pre skupiny bitov, ktoré sú usporiadané ako "rez" v slove, je možné vykonať recykláciu hodnôt niekoľkokrát cez vnútorné obvody ešte pred tým, ako sa na výstup odovzdá konečný výstupný zoznam. Počas recyklácie je na výstupe momentálny výstupný zoznam, ktorý ale môže byť použitý ako momentálny vstupný zoznam.

Zoznam môže obsahovať aplikáciu funkcie spoločne s hodnotami, ktoré majú byť spracované, alebo iný typ zoznamu ako

zoznam, ktorý obsahuje aspoň jednu hodnotu typu integer alebo hodnotu s plávajúcou rádovou čiarkou. Keď obsahuje aplikáciu funkcie, zoznam prvých prvkov, ktorý je na in/output NU_0 , obsahuje inštrukčný kód, teda informáciu o druhu spracovania, ktoré sa má vykonať a zvyšok prvkov zoznamu sú argumenty k inštrukcii.

Inštrukcia je vykonaná prepísaním inštrukcie v jednom alebo niekoľkých krokoch, aby sa prípadne získal výsledok. Mikroinštrukcie a stavové signály sú používané vo vnútri číslicovej ALU. Používané sú samostatnou riadiacou jednotkou 1a vo vnútri číslicovej ALU na riadenie poľa 2a. Keď sa prepisuje priamo do výsledku, hodnota je umiestnená vo výstupnom zozname. Keď sa vykonáva prepisovanie po kroku, zoznam je prepísaný tak, aby v ňom bol uložený rovnaký inštrukčný kód alebo nový alebo ľahko pozmenený inštrukčný kód a aby v ňom bol uložený inštrukčný kód v mnohých prípadoch nasledovaný hodnotou alebo hodnotami. Je nutné si všimnúť, že najlepší režim pre tvorbu operácie prepísania v niekoľkých cykloch je udržiavať kód inštrukcie intaktný v priebehu celej operácie, pretože riadiaca jednotka 6 bude vtedy menej zložitá.

Interfejs číslicová ALU - riadiaca jednotka.

Keď sa vrátíme k obr. 24A, hodinový signál C_{ALU} sa používa pre generovanie hodinových impulzov pre číslicovú ALU a číslicová ALU určuje, kedy je pripravená prijať hodinové impulzy sa signálom ready na výstupe A_{BUSY} .

Obsah častí hlavných registrov umiestnených v rovinách HEAD, teda príznakové slová sú privedené na riadiacu jednotku 6, ktorá potom riadi číslicovú ALU riadiacej jednotky 1a.

Súčasťou vynálezu sú príznaky obsiahnuté v informácii, ktorá sa odovzdáva na vstup/výstup tak, že informácia o príznaku je privedená priamo na centrálnu jednotku, ďalej tiež nazývanú číslicová ALU riadiaca jednotka.

Riadiaci vstup A_{CNTL} sa používa na riadenie číslicovej operácie číslicovej ALU. Operácia môže trvať niekoľko hodinových cyklov.

Výstup S_{STATE} je vrátený do riadiacej jednotky 6 po každej

operácii. Inštrukčný kód na výstupe A_{INS} je zhustenie inštrukčného kódu v hlavnom registri, ku ktorému je zbernica MU_0 pripojená.

Podľa obr.24B základný hodinový kmitočet pre procesor z riadiacej jednotky 6 je C_{10} , ktorý má napríklad periódu 10 ns. Operácia číslicovej ALU je privedená v jednom alebo niekoľkých takých hodinových cykloch. Jadrový procesor, ktorý je pripojený na číslicovú ALU, by mohol byť riadený s periódou, ktorá nie je dlhšia ako napríklad dvojnásobok periódy pre číslicovú ALU, teda s hodinovým signálom C_{20} , ktorý má periódu 20 ns pokiaľ C_{10} má periódu 10 ns. Miesto toho je však možné riadiť ho periódou, ktorá je kratšia. Základný riadiaci signál C_{ALU} pre zariadenie, ktoré má kratšiu periódu (teda číslicová ALU v uvedenom zapojení alebo inak jadrový procesor), je preto natiahnutý na konci inštrukcie, aby bol synchronizovaný s hodinovým signálom C_{20} , ktorý má dlhšiu periódu. Číslicová ALU vracia signál vo vysokej úrovni A_{BUSY} , pokiaľ potrebuje viac hodinových cyklov typu C_{10} na vykonanie inštrukcie. Natiahnutie sa vykoná potom, keď signál A_{BUSY} má nízku úroveň, ako je zrejmé z obr.24B.

Reprezentácia hodnôt.

Číslicová ALU má premenné delenie medzi exponentom a mantisou v reprezentácii s plávajúcou rádovou čiarkou. Potom je možné mať pre malé čísla mnoho bitov v mantise, čo dáva výhodu vysokej presnosti. Na druhej strane je možné pracovať s extrémne veľkými číslami.

Ako je schématicky ukázané na obr.25, každý bitový vzorec na zbernici, ktorý má číselnú hodnotu má preto krátke bitové pole v najvýznamnejších bitoch bitového reťazca, kde toto pole nesie čiastočne informáciu o hodnote znamienka a čiastočne o znamienku exponenta a čiastočne kódu, ktorý berie do úvahy polohu delenia (pravá šípka na obr.25) medzi časťou exponentu e a mantisy m číselnej hodnoty. Je však nutné si všimnúť, že dĺžka kódovej časti bitového poľa sa môže tiež meniť a že poloha delenia medzi kódovým poľom a poľom exponentu je určená kódovým poľom, ako je

znázornené ľavou šípkou na obr.25.

Číslíková ALU realizuje reprezentáciu čísla s plávajúcou rádovou čiarkou rovnakým spôsobom ako reprezentácia typu integer pre rovnaké číslo. Preto hodnota v plávajúcej rádovej čiarkke a hodnota pre integer rovnakého bitového vzorca je úplne odlišná.

Tento cieľ však môže byť splnený, keď budeme mať interpretáciu vzorca pre bitový reťazec, pokiaľ reprezentácia má byť interpretovaná ako hodnota s plávajúcou rádovou čiarkou a druhú interpretáciu vzorca pre bitový reťazec, pokiaľ reprezentácia má byť interpretovaná ako hodnota typu integer. Takto žiadna z reprezentácií nemusí byť v binárnom tvare, namiesto toho aspoň jedna, lepšie však obidve, je v kódovanom tvare. Transformácia bitového vzorca sa vykoná automaticky a hardware v počítači vykoná výpočet na transformovaných bitových vzorcoch implicitne podľa nasledujúcich pravidiel, teda bez znalosti toho, aká interpretácia bola vytvorená.

Je snaha vykonať najbližšiu reprezentáciu hodnoty v plávajúcej rádovej čiarkke, teda že tu nebudú interpretované dva bitové vzorce ako rovnaké číslo.

Grafická reprezentácia typu integer.

Na obr.26A je uvedená grafická reprezentácia kódu pre hodnotu typu integer podľa vynálezu hodnoty, nazývanej integer H. Binárna reprezentácia sa znázorňuje pozdĺž horizontálnej súradnice a hodnoty H pozdĺž vertikálnej súradnice. Najvýznamnejší bit predstavuje znamienko hodnoty, či je kladné alebo záporné. Ako je zrejmé z obr.26A, číslo H má najväčšiu zápornú hodnotu keď binárna reprezentácia sú samé "0", má nulu uprostred binárnej reprezentácie, teda keď najvýznamnejší bit sa zmení z "0" na "1" a má svoju najväčšiu kladnú hodnotu, keď binárna reprezentácia sú samé "1", teda má svoju maximálnu binárnu hodnotu $\max_{rep}$.

Grafická reprezentácia typu plávajúcej rádovej čiarky.

Na obr.26B je uvedená grafická reprezentácia kódu pre plávajúcu rádovú čiarku podľa vynálezu, ktorá sa nazýva hodnoty H

s plávajúcou rádovou čiarkou. Binárna reprezentácia je umiestnená na horizontálnej súradnici a hodnoty H sú umiestnené na vertikálnej súradnici, čo je tu vedené bodom, v ktorom hodnoty H s plávajúcou rádovou čiarkou menia znamienko. V tejto reprezentácii najvýznamnejší bit predstavuje znamienko hodnoty. Hodnota s plávajúcou rádovou čiarkou musí splniť požiadavku na univerzálnosť. Pre určitú presnosť musí byť magnituda hodnoty ľubovoľne premenná. Takto je možné vyjadriť extrémne malé a veľké hodnoty. Merítka pre hodnoty H s plávajúcou rádovou čiarkou je logaritmické. Logaritmická hodnota H môže byť zjednodušená použitím lineárnych aproximácií s každým činiteľom 2. Výsledok je typický kód pre plávajúcu rádovú čiarku. Ako je vidieť z obr. 26B, binárna reprezentácia hodnôt H medzi -1 a $+1$ sa nachádza v strede poľa binárnej reprezentácie a zaberá polovicu poľa.

Relatívna presnosť hodnoty je konštanta cez celý rozsah hodnôt. Logaritmické hodnoty majú určitú presnosť a rozsah. Pokiaľ by magnituda mala byť ľubovoľne veľká, musí byť možné nastaviť rozsah logaritmu.

Logaritmická hodnota môže byť aplikovaná na logaritmickú funkciu raz alebo aj viackrát. Takto je možné vyjadriť veľmi veľké hodnoty. Presnosť sa však pri veľkých hodnotách znižuje, ale presnosť u malých hodnôt je možné udržať alebo dokonca trochu zvýšiť.

Pretože sa dva bitové vzorce dvoch reprezentácií objavujú rovnakým spôsobom, komparátor hodnoty typu integer môže byť použitý tiež pre reprezentáciu hodnoty v plávajúcej rádovej čiarkke. To znamená, že komparátor pre konkrétnu hodnotu s plávajúcou rádovou čiarkou môže byť vynechaný. To na druhej strane znamená, že komparátor veľkosti typu integer, ktorý vykonáva komparáciu iba v jednom cykle, môže byť použitý pre hodnoty s plávajúcou rádovou čiarkou, čo je veľká výhoda. Komparátor hodnoty s plávajúcou rádovou čiarkou konvenčného typu musí použiť mnoho cyklov pre túto operáciu.

V konvenčnej aritmeticko-logickej jednotke je možné interpretovať niekoľko bitových vzorcov ako rovnaké číslo a je

vykonaná normalizácia tak, že je zvolený jeden z týchto bitových vzorcov. Podľa vynálezu nie je nutné vykonávať žiadnu normalizáciu aspoň nie pre aktuálny výpočet. Normalizáciu je možné vykonávať pred a po operácii výpočtu. Keď máme tesnú podobnosť reprezentácie medzi hodnotami s plávajúcou rádovou čiarkou, výpočet hodnoty s plávajúcou rádovou čiarkou, napríklad komparácia môže byť určená iba v jednom cykle. Výhoda tesnej podobnosti reprezentácie je tiež v tom, že je k dispozícii maximálny počet hodnôt s plávajúcou rádovou čiarkou, teda v rade hodnôt nie sú žiadne vylúčené alebo nepoužité hodnoty.

Dôvod, prečo tu je niekoľko bitových vzorcov, ktoré môžu byť interpretované rovnakým spôsobom v konvenčnej aplikácii je ten, že exponenciálna časť a časť s mantisou hodnoty sú násobené medzi sebou, aby sa vytvorila hodnota. Napríklad hodnota 1,0 môže byť $2^{1 \cdot 1/2}$, teda že bitový vzorec exponentu je 01 a bitový vzorec mantisy je 100, alebo $2^{2 \cdot 1/4}$, teda že bitový vzorec exponentu je 10 a bitový vzorec mantisy je 010, pričom bitový vzorec exponentu a bitový vzorec mantisy nasledujú po sebe.

Tesná reprezentácia sa vytvorí implicitným odovzdaním "1", teda pre bitovým vzorcom mantisy pre kladné hodnoty a "0" implicitne pre záporné hodnoty. Mantisa je klasicky špecifikovaná ako hodnota medzi -1 a +1. Binárne kódovaná mantisa je špecifikovaná ako hodnota v rozsahu (1,0 až 2,0) alebo (-2,0 až -1) v závislosti na znamienku.

Kódovanie reprezentácie.

Vzorcie integer sú presne určené a reprezentácie plávajúcej rádovej čiarky budú mať rovnaký konsekutívny rád ako interpretácia pre typ integer, čo je veľmi tvrdá požiadavka.

Výhodné riešenie tohto problému je schématicky uvedené na obr. 26C. Obr. 27A až 27D ukazujú datový zoznam slov, ktorý je vytvorený podľa princípov na obr. 26C. Na obr. 26C je uvedená schématická grafická ukážka kódu s plávajúcou rádovou čiarkou, binárna reprezentácia je uvedená na ľavej strane priamky LIN vertikálneho rozdelenia a hodnoty H s plávajúcou rádovou čiarkou sú uvedené na

pravej strane priamky LIN. Čiara LIN je znázornená ako os, ktorá ide smerom k vyšším hodnotám. To je dané ako binárnou reprezentáciou tak i reprezentáciou hodnoty H. Reprezentácia a hodnoty H sú znázornené tak, že von postupujú po oboch stranách priamky LIN. Najbližšie k priamke LIN je znázornenie súradnic na grafe na obr. 26B - súradnica hodnoty H napravo a súradnica binárnej reprezentácie naľavo. Prvé pole mimo súradnice, sh napravo a s naľavo predstavuje najvýznamnejší bit. Ten je "0" pre záporné hodnoty a "1" pre kladné. Druhé pole, seh a se, predstavuje ďalší najvýznamnejší bit. Je to znamienkový bit exponentu. Tretie pole obsahuje minimálne jeden bit, ktorý má menší význam ako ďalší najvýznamnejší bit, jeden bit pokiaľ binárne pole je nízke ako je ukázané v prípade s iba siedmymi bitmi a viac ako jeden bit pokiaľ binárne pole je vysoké, ako bude ukázané ďalej pre prípad s 32 bitmi.

V príklade na obr. 26C, 27A až 27D je uvedené kódovanie pre slová so siedmimi bitmi namiesto 32-bitových slov, ktoré bude pravdepodobne v praxi používané oveľa častejšie. Binárna reprezentácia hneď naľavo od stredovej priamky bude interpretovaná s hodnotami H -1024 až +768. Je nutné si všimnúť, že hodnoty H -1 a +1 sú umiestnené v polovici medzi hodnotou 0 a extrémnymi hodnotami -1024 a +768. Prvý binárny bit a a sh pre reprezentáciu i pre hodnoty H je znamienko hodnoty, či je kladná alebo záporná.

Druhý binárny bit je znamienko exponentu, teda či absolútna hodnota je nad alebo pod číslom 1. Znamienko exponentu bitu H seh je preto kladné medzi -1024 a -1, záporné medzi -1 a +1 a potom kladné medzi +1 a +768. Ako je zrejmé z ľavej strany obr. 26C, znamienko reprezentácie exponentu se je namiesto toho záporné v prvej štvrtine binárneho radu, kladné v druhom, záporné v treťom a kladné vo štvrtom. Bit seh môže byť popísaný tak, že keď $s = 1$, potom $seh = se$, ináč $seh = 1 - se$.

Preto v reprezentácii najvýznamnejší bit, ktorý predstavuje znamienko znaku, má najpomalšiu zmenu, ďalší najvýznamnejší bit, ktorý predstavuje znamienko exponentu, má najpomalšiu zmenu,

teda dvakrát takú, ako pre znamienkový bit. Bit na ďalšom mieste má dvakrát takú zmenu ako pre bit exponentu a je to kódový bit c . Kód v príklade podľa obr. 26C a 27A až 27D obsahuje iba jeden bit. V ostatných prípadoch môže kódové pole obsahovať viac ako jeden bit. Kód riadi dĺžku exponentu a je nutné na neho pozerieť ako na predchádzajúce znamienkové bity.

Hodnota H s plávajúcou rádovou čiarkou ktorá sa má odovzdať, teda zápis v plávajúcej rádovej čiarky vytvára kódové pole ch konkrétnejším spôsobom. Ako je zrejme z časti eh ktorá dáva hodnotu H exponentu priamo, hodnota eh ide z veľkej kladnej hodnoty dole až na hodnotu 0, kde hodnota H je -1, odtiaľ na veľkú zápornú hodnotu, kde je hodnota H 0, odtiaľ na hodnotu 0, kde hodnota H je +1 a odtiaľ na veľkú kladnú hodnotu, kde hodnota H je 768. Časť eh_{abs} ukazuje absolútnu hodnotu H exponentu a je reprezentovaná dvomi krivkami, ktoré prechádzajú vedľa seba, obidve majú konce na veľkej kladnej hodnote a ich stredový bod je umiestnený na 0. Kódové pole pre hodnotu H je zostavené s ohľadom na hodnotu H eh_{abs} a je znázornené tak, že má rovnaký tvar v čiarkovanej čiare a tento tvar bude mať, keď kódové pole bude obsahovať iba jeden bit v súvislých priamkach. Preto $ch = c$, ak $se = 1$ a inak $ch = 1 - c$, keď kód ch obsahuje iba 1 bit.

Je jednoduché sledovať usporiadanie zoznamu, ktorý je na obr. 27A až 27D, pretože každé z bitových polí s , sh , se , seh , c a ch obsahuje iba jeden bit a preto komparácia medzi reprezentáciami na obr. 26 a obr. 27A až 27D je zrejmá. Prvý stĺpec na obr. 27A až 27D je sekvenčný binárny zoznam od 000000 do 1111111 a jeho priamy protiklad v dekadickom číselnom systéme je v druhom stĺpci. Napravo od stĺpca so šípkou sú ukázané transformované čísla pre reprezentáciu a hodnoty H .

V zozname sú súvislé horizontálne priamky pre každý spínač reprezentačného kódového poľa, aby sa ľahko vykonávala komparácia s ilustráciou na obr. 26C.

Stĺpec napravo od ch obsahuje dekadickú reprezentáciu hodnoty exponentu e ďalej dekadickú reprezentáciu hodnoty exponentu ch hodnoty H . Ďalšie dva stĺpce ukazujú binárnu

reprezentáciu hodnôt e a eh . Ďalšie dva nasledujúce stĺpce ukazujú dekadické a binárne reprezentácie hodnoty mantisy m . Po nich je ukázaná dekadická reprezentácia hodnoty H . Je nutné si všimnúť, že mantisa sa mení medzi -1 a -2 na obr. 27A a 27B, kde hodnota sh je "0" a medzi 11 a $+2$ na obr. 27C a 27D, kde hodnota sh je "1". Stĺpec ktorý je najviac vpravo ukazuje dekadickú hodnotu H v binárnej kombinácii po transformácii zo stĺpca najviac vľavo.

Ako je zrejmé z hornej časti obr. 27C, reprezentácia ktorá má "1" ako svoj najvýznamnejší bit po ktorej nasleduje "0":s, predstavuje konkrétnu hodnotu "0" v plávajúcej rádovej čiarky.

Ako je zrejmé zo sekvencie binárnych čísiel na obr. 26C, naľavo od stredovej priamky sedem binárnych bitov zdieľa svoje najmenej významné bity naľavo s exponentom a mantisou, pretože tri najvýznamnejšie bity sú rezervované pre znamienkové bity a kódový bit. Z číslicových stĺpcov e a m je zrejmé, že v najvyššej časti, kde znamienko exponentu se je záporné, teda "0", a kódová hodnota c je nula, hodnota e obsahuje tri bity a teda hodnota m jeden bit (prvé dve "0":s v stĺpci môžu byť vynechané). To je uvedené na obr. 26C, kde e -reprezentácia je v tejto časti v mnohých krokoch. V ďalšej časti, kde znamienko exponentu je stále záporné, ale kódová hodnota c je "0", hodnota e neobsahuje iba jeden bit (prvé dva "0":s v stĺpci môžu byť vynechané) a hodnota mantisy môže obsahovať tri bity. To je uvedené na obr. 26, kde má e -reprezentácia v tejto časti málo krokov. V ďalšej časti hodnota e obsahuje iba jeden bit a mantisa tri bity, atď.

Ako je uvedené v časti reprezentácie, exponent e má veľa krokov v častiach, ktoré majú $c = 0$ a $se = 0$ a málo krokov v častiach, ktoré majú $c = 1$ a $se = 1$. V oddiele e +bázy naľavo od oddielu e sú znázornené súvislými čiarami hodnoty e , uvedené za sebou v každej štvrtine celého oddielu a čiarkovanou čiarou, ako sú hodnoty eh_{abs} vytvárané invertovaním každej druhej štvrtiny oddielu v e +bias.

Preto je kódový bit alebo bity závislý na predchádzajúcom znamienkovom bite exponentu a preto je možné povedať, že kód obsa-

huje v skutočnosti znamienkové bity a preto môže mať obmedzenú dĺžku, to znamená kratšiu ako keby dĺžka exponentu bola závislá od toho, čo je uvedené iba v kódovom poli.

Dĺžka mantisy závisí na hodnote exponentu tak, že keď je exponent dlhý, mantisa je krátka a naopak. Preto mantisa na obr. 26 nie je uvedená, ale vyplýva zo stĺpca m , m zoznamu na obr. 27A až 27D, kde prvé m udáva dekadický údaj hodnoty a druhé m udáva jeho číselný protiklad. Ak $s = 1$ potom $mh = 1.0 + m$, ináč $mh = -2 + m$, ako bolo uvedené.

Horeuvedený príklad bol popísaný pre slovo o dĺžke sedem bitov pre účely čitateľnosti. Ako bolo uvedené, dĺžka slova môže byť vybraná tak, že bude celkom dlhá, napríklad 32 bitov. Potom je praktické mať premennú dĺžku kódu. Je tiež praktické, aby bol obvod menej zložitý, zmeniť bod delenia medzi exponentom a mantisou v krokoch niekoľkých bitov, teda štyri bity. Výhodou pri tom je, že nie je nutné mať toľko kódov, koľko by muselo byť, keď zmena delenia je vykonaná po bite. Príklad takého dekodéru presnosti je popísaný ďalej.

Použitie číslicovej ALU.

Na obr. 28 je podrobné zapojenie číslicovej ALU, ktorá môže byť zahrnutá v procesore prvého rádu podľa vynálezu. Vstupné/výstupné zbernice NU_0 až NU_3 sú pripojené na vstupnú/výstupnú vyrovnávaciu pamäť 20, ktorá prenáša informáciu na vstup interných vodičov v_{01} až v_{31} pre každý invertovaný hodinový interval externého hodinového signálu CLOCK, ktorý je privedený na hodinový riadiaci vstup vyrovnávacej pamäte a výstup na interných výstupných zberniciach v_{0out} až v_{3out} na vstupné/výstupné zbernice NU_0 až NU_3 pre každý hodinový interval.

Zapojenie obvodu 20 a jeho taktovanie je uvedené na obr. 29A, 29B a 29C. Vstupná/výstupná vyrovnávacia pamäť 20 obsahuje vstupné vyrovnávacie pamäte, jednu pre každý vstup/výstup zbernice NU_0 až NU_3 a rovnaký počet výstupných vyrovnávacích pamätí. Aby sa zabránilo problémom predháňania, hodinové signály sa nesmú prekrývať. To je docielené tým, že sa ľahko orežú ich nábežné

hrany.

V priebehu prvej fáze hodinového signálu CLOCK, ktorý je uvedený na obr. 29A a ktorý je invertovaný, ako je označené naľavo zátvorkami, vstup z hlavných registrov v jadrovej bunke je privedený na internú dátovú cestu v číslicovej ALU cez vstupné vyrovnávacie pamäte, ktoré sú riadené do stavu vysielania. Dáta potom pokračujú cez internú dátovú cestu v číslicovej ALU a sú pritom prevedené do ďalšieho stavu na výstupných zberniciach. Výstupné vyrovnávacie pamäte sú riadené do blokovacieho stavu.

Ako je uvedené na obr. 29B, počas fázy 1 hodinovej periódy hodinového signálu CLOCK, označené na obrázku zátvorkami naľavo, vstupné vyrovnávacie pamäte v obvode 20 sú riadené do blokovacieho stavu a interné obvody v číslicovej ALU pracujú ako stabilný multiplexor pre riadenie signálov, ktoré majú byť privedené na vstupné/výstupné zbernice NU_0 až NU_3 . Výstupné vyrovnávacie pamäte v obvode 20 riadené do stavu vysielania.

Podmienka pre inštrukciu s viac cyklami vo fáze 0 je uvedená na obr. 29C. Obsah vstupných/výstupných zberníc NU_0 až NU_3 , ktorý je na ne privedený z číslicovej ALU v priebehu predchádzajúcej fázy 1 je na nich pridržený v priebehu fázy 0 z dôvodu kapacitných vplyvov a takto je možné ho prenášať cez vstupné vyrovnávacie pamäte, ktoré sú riadené do stavu vysielania, zatiaľ čo výstupné vyrovnávacie pamäte sú riadené do stavu blokovania na vstupných zberniciach v_{01} , v_{11} , v_{21} , v_{31} a odtiaľ je odovzdaný ďalej cez internú dátovú cestu v číslicovej ALU, keď je prenesený do ďalšieho stavu. Vstupný zoznam je použitý pre riadenie a vykonanie výpočtu, ktorý je vykonaný prepísaním obsahu vo vstupnom zozname a k preneseniu výsledku na výstup. Prvý prvok v zozname je inštrukčný kód a zvyšok argumentov prvku pre inštrukciu, ktorá sa vyžaduje, aby funkčný hardware bol závislý na vstupných dátach. Ale informácia, ktorá má tvar podľa druhu informácie, ktorá je prítomná na zbernici v_{11} , v_{21} , v_{31} je daná na vstup EXT z riadiacej jednotky 6.

Riadiaca jednotka 6 má informáciu o tom, či slová na vstupných zberniciach v_{01} , v_{11} , v_{21} , v_{31} predstavuje inštrukcia

a/alebo čísla a či čísla sú typu plávajúcej rádovej čiarky alebo integer a túto informáciu odovzdá číslicovej ALU v obvode 27 cez zbernicu EXT.

Zapojenie nasledujúcich jednotiek spracovania je uvedené na obr.28 a vykonáva nasledujúce funkcie:

Komparátory..

Prvý komparátor 21 je zapojený na vodiče v_{01} a v_{11} , kde sa vykonáva komparácia celej informácie týchto vodičov, to znamená, že vykonáva $v_{01} \quad v_{11}, v_{01} = v_{11}$ a odovzdáva výsledky komparácie teda dvojbitovú hodnotu na výstup cmp_{01} , ktorý je pripojený na riadiaci obvod 27 číslicovej ALU. Druhý komparátor 22 je pripojený na vodiče v_{11} a v_{21} a vykonáva komparáciu celej informácie na týchto vodičoch, to znamená, že vykonáva $v_{11} \quad v_{21}, v_{11} = v_{21}$ a výsledok komparácie odovzdá na výstup cmp_{12} , ktorý je pripojený na riadiaci obvod 27. Tretí komparátor 221 je pripojený na vodiče v_{21} a v_{31} , to znamená, že vykonáva $v_{21} \quad v_{31}, v_{21} = v_{31}$ a výsledok komparácie odovzdáva na výstup cmp_{23} , ktorý je pripojený na riadiaci obvod 27. Štvrtý komparátor exponentu 222 je pripojený na vodiče v_{11} a v_{21} , každý je zbernica o dĺžke 32 bitov a komparuje vodiče najvýznamnejšiu časť slova s týmito vodičmi, teda časti se, c, e, teda všetky informácie okrem mantisy a znamienka na týchto vodičoch. Komparátor si všíma kódovanie častí se, c, e, ktoré už boli popísané. Preto signál B_{1sb1} z dekodéru presnosti PD je privedený na samostatný vstup komparátora 222.

Piaty komparátor 223 vykonáva komparáciu signálov B_{1sb1} a B_{1sb2} z dekodéru presnosti. Výsledný signál cmp_prec udáva, či signály B_{1sb1} a B_{1sb2} sú zhodné.

Riadiaci obvod 27 odovzdáva výsledok komparácie z komparátorov na svoj výstup A_{STATE} , pripojený na riadiacu jednotku 6 (obr.24A). Výsledok môže byť tiež použitý na riadenie niektorých jednotiek, ktoré budú popísané ďalej tak, že dopredu vybrané slová sú odovzdané na výstupných zberniciach v_{1out}, v_{2out} a v_{3out} .

Bezvýznamné dekodéry.

Bezvýznamný dekodér 23 je pripojený na vodič v_{11} a testuje, či bity na V_{11} sú všetky "0" alebo "1" v prípade, že slovo na zber-

nici v_{11} reprezentuje integer a keď slovo na zbernici reprezentuje typ s plávajúcou rádovou čiarkou, testuje samostatne či bity v exponenciálnej časti a v časti mantisy sú všetky "0" alebo "1". Signál B_{1sb1} , ktorý udáva najmenej významnú skupinu bitov vstupu v_{11} , dáva informáciu o tom, či slovo na vstupe zbernice predstavuje hodnotu typu integer alebo s plávajúcou rádovou čiarkou a v prípade plávajúcej rádovej čiarky udáva rozdelenie na časť exponentu a mantisy. Výsledok týchto testov sa objaví na výstupe $insig_1$, ktorý je pripojený na vstup riadiaceho obvodu 27. Bezvýznamný dekodér 24 je pripojený na vodič v_{21} a testuje, či bity na v_{21} sú všetky v "0" alebo "1", keď slovo na zbernici v_{21} predstavuje typ integer a pokiaľ slovo na zbernici v_{21} predstavuje typ s plávajúcou rádovou čiarkou, testuje samostatne, či bity v exponenciálnej časti a v časti mantisy sú všetky v "0" alebo v "1". Signál B_{1sb2} udáva najmenej významnú skupinu bitov v_{21} a dáva rovnaký druh informácie ako signál B_{1sb1} . Výsledok týchto testov je odovzdaný na výstup $insig_2$, ktorý je pripojený na vstup riadiaceho obvodu 27. Bezvýznamný dekodér 25 je pripojený na vodič v_{31} a testuje, či slovo na v_{31} sú samé "0" alebo "1" reprezentuje hodnotu integer alebo pokiaľ slovo na zbernici v_{31} reprezentuje typ s plávajúcou rádovou čiarkou, testuje samostatne, či bity v exponenciálnej časti a časti mantisy sú všetky "0" alebo "1". Informácia, ktorá sa toho týka, je odvodená od signálu B_{1sb1} . Výsledok týchto testov je odovzdaný na výstup $insig_3$, ktorý je pripojený na vstup riadiaceho obvodu 27. Každý bezvýznamný dekodér je tiež riadený riadiacim signálom $c5_3$ z riadiacej jednotky 27, ktorý bude popísaný ďalej.

Dôvodom zavedenia bezvýznamných dekodérov je to, aby sa zistilo, či informácia na jednom zo vstupov, ktorá sa zúčastní vykonávaného výpočtu, by mala byť bezvýznamná. Táto informácia je tiež odovzdávaná na zbernici A_{STATE} do riadiacej jednotky 6. Finálna podmienka je, že minimálne jeden vstup z jednej bezvýznamnej jednotky napríklad udáva, že riadiaca jednotka 27 pre niektoré typy operácií môže použiť všetky nuly.

Dekodér inštrukcie.

Dekodér inštrukcie 26 je pripojený na vodič v_{01} . Zbernica v_{01}

prenáša informáciu, kedy má číslicová ALU vykonať aritmetickú operáciu. Výsledok dekódovania v_{01} je odovzdaný do riadiacej jednotky 27 cez výstup ins , či v_{01} obsahuje inštrukciu alebo dátovú hodnotu. Jednotky 23 až 26 sú tiež riadené ostatnými signálmi B_{1sb1} , B_{1sb2} a c_5 , ktoré sú generované v obvodoch PD1, PD2 a 27 a sú popísané ďalej.

Sčítačka a odčítačka.

Prvá sčítačka 28 pripojená na vodiče v_{1i} a v_{2i} vykonáva súčet $v_{1i} + v_{2i}$ a výsledok odovzdá na oddelený výstup a_1 . Možný signál prenosu výsledku bude privedený na výstup gr_{a1} , ktorý je pripojený na riadiaci obvod 27. Odčítačka 29, pripojená na vodiče v_{1i} a v_{2i} vykonáva rozdiel $v_{1i} - v_{2i}$ a výsledok odovzdá na oddelený výstup a_2 . Možný signál prenosu výsledku bude privedený na výstup gr_{a2} , ktorý je pripojený na riadiaci obvod číslicovej ALU. Druhá sčítačka 30, pripojená na vodiče v_{1i} vykonáva súčet $v_{1i} + 2 \cdot v_{2i}$ a výsledok odovzdá na oddelený výstup a_3 . Možný signál prenosu výsledku bude privedený na výstup gr_{a3} , ktorý je pripojený na riadiacu jednotku 27. Každá sčítačka je tiež riadená signálom c_5 z riadiaceho obvodu 27 a signálom B_{1sb1} . Tento signál dáva informáciu, či slová na vstupných zberniciach predstavujú hodnoty typu integer alebo s plávajúcou rádovou čiarkou a v prípade plávajúcej rádovej čiarky a porovnaných hodnôt na zbernici v_{1i} a v_{2i} dáva rozdelenie medzi exponenciálnu časť a časť mantisy. Tieto informácie sú podstatné, pretože diskkrétne hodnoty, teda integer sú kompletne vyhodnotené, ale pre súvislé hodnoty, teda s plávajúcou rádovou čiarkou sú exponenciálna časť a časť mantisy vyhodnotené samostatne.

Obvody znamienka.

Prvý obvod 35 znamienka a znamienka exponentu triedi znamienko, znamienko exponentu, najvýznamnejší bit mantisy a dva najmenej významné bity mantisy slova na zbernici v_{1i} a odovzdá svoj výstup $sign_1$ na vstup riadiaceho obvodu 27. Druhý obvod 36 znamienka a znamienka exponentu triedi znamienko, znamienko exponentu, najvýznamnejší bit mantisy a dva najmenej významné bity mantisy slova na zbernici v_{2i} a odovzdá svoj výstup $sign_2$ na

vstup riadiaceho obvodu 27. Tiež tretí obvod 37 znamienka a znamienka exponentu môže byť pripojený na zbernicu v_{3i} a odovzdáva svoj výstup $sign_3$ na vstup riadiaceho obvodu 27. Je nutné si tiež všimnúť, že obvody 35 a 36 sú nutné, keď bit znamienka, znamienka exponentu a mantisy sú odovzdané ako časť slova, ktoré je na zbernici v_{1i} a v_{2i} . Používané sú číslicovou ALU v prípade slov s plávajúcou rádovou čiarkou na vstupných zberniciach. Obvod 37 môže byť v niektorých aplikáciách číslicovej ALU vynechaný. Obvod znamienka a znamienka exponentu (nie je uvedené) môže byť tiež v niektorých aplikáciách číslicovej ALU k dispozícii pre zbernicu v_{0i} .

Riadiaci obvod číslicovej ALU.

Riadiaci obvod 27 dáva číslicový výstupný signál na výstupných zberniciach c_1, c_2, c_3, c_4, c_5 tým, že spracuje jeho vstupné signály. Riadiaci obvod 27 je s výhodou boolovské hradlové pole. Príklady riadenia, ktoré sú realizované výstupnými signálmi, budú popísané neskoršie.

Jednotky operandu.

Dva operačné jednotky r_1 a r_2 , jedna pre každý operand na vodičoch v_{1i} a v_{2i} , obsahujú všetku aritmetiku, rovnakú pre jeden prvok. Každá jednotka operandu má niekoľko vstupov informácie. Tri z nich sú pripojené na samostatné výstupy a_1, a_2 a a_3 troch jednotiek 28, 29 a 30, dve z nich na zbernici v_{1i} a v_{2i} a tri z nich sú pripojené k niektorým výstupom jednotlivých dekodérov presnosti PD1 alebo PD2, ktoré budú popísané ďalej. V princípe každý vstup informácie je zbernica, ktorá má rovnaký počet vodičov ako každá vstupná zbernica $v_{0i}, v_{1i}, v_{2i}, v_{3i}$.

Výstup jednotky operandu r_1 je výstupná zbernica v_{1out} a výstup jednotky operandu r_2 je výstupná zbernica v_{2out} .

Jeden z výstupov c_1 riadiaceho obvodu 27 je kompozitný riadiaci signál na riadenie vnútorných prvkov v jednotke operandu r_1 . Ostatné riadiace signály pre jednotku r_1 sú odovzdávané dekodérom presnosti PD1. Druhý z výstupov c_2 riadiacej jednotky 27 je kompozitný riadiaci signál pre riadenie vnútorných prvkov v jednotke operandu r_2 . Ostatné riadiace signály pre jednotku r_2 sú

odovzdávané dekodérom presnosti PD2. Tretí výstup c_3 je kompozitný signál, ktorý sa prenáša do tretej jednotky operandu r_3 a štvrtý výstup c_4 sa prenáša na riadenie polynómu, ktorý dáva obvod 31 a bude popísaný neskoršie, piaty výstup c_5 riadi dekodéry presnosti PD1 a PD2 a tiež bezvýznamné obvody 23 až 25, sčítačky 28, 30 a odčítačky 29 a obvody znamienka a znamienka exponentu 35 až 37.

Každá jednotka operandu r_1 alebo r_2 má v podstate rovnakú konfiguráciu a preto bude podrobne popísaná iba r_2 . Rovnaké prvky v jednotkách r_1 a r_2 majú rovnaké referenčné označenia. Jednotka operandu je rozdelená na dve skupiny, časť exponentu OP_e a časť mantisy OP_m . Časť mantisy OP_m má výstup n_1 a časť exponentu má výstup n_2 . Výstupy n_1 a n_2 sú pripojené na vstupy selektora výstupnej zbernice, nazývanej tiež individuálny skupinový selektor M1. Generátor 0/1 40, ktorý dáva skupinu presnosti štyri "0" alebo "1" podľa voľby, ktorá je riadená riadiacim signálom c_{211} do generátora, je tiež pripojený na samostatný vstup selektora M1.

Exponenciálna časť OP_e má ako vstupy štyri zo vstupov jednotky operandu, teda v_{2i} a B_{bias2} , B_{co_incr2} , B_{co_decr2} z dekodéru presnosti PD2 a tiež má vstup n_8 z časti mantisy OP_m . Vstup v_{2i} je pripojený na vstup inkrementačného člena INC, vstup dekrementačného člena DEC a na jeden z piatich vstupov selektora slova M2. Výstupy n_5 a n_6 komponentov INC a DEC sú pripojené na dva vstupy selektora slova M2. B_{co_incr2} , B_{co_decr2} sú pripojené na posledné dva vstupy selektora slova M2.

Zložka INC v jednotke operandu r_2 má tiež výstupy pre prenos gr_{i2} a ce_{i2} , zatiaľ čo zložka DEC má výstupy pre prenos gr_{d2} a ce_{d2} pripojené na riadiacu jednotku 27. Tiež jednotka operandu r_1 má výstupné signály prenosu inkrementačnej zložky gr_{i1} a ce_{i1} a výstupné signály prenosu dekrementačnej zložky gr_{d1} a ce_{d1} pripojené na riadiacu jednotku 27. Výstupy prenosu sú prenesené hneď v priebehu prvej polovice hodinového cyklu. Selektory sú riadené v priebehu prvej polovice hodinového cyklu a potom riadenie tohoto selektora sa stará o to, či v niektorom obvode inkrementácie alebo dekrementácie prišlo k prenosu alebo nie.

Normálne maximálne oneskorenie k prvému selektoru je väčšie

ako polovica cyklu. Preto v tomto usporiadaní už nie sú žiadne ďalšie oneskorenia. Typické oneskorenie je však obecné menšie ako polovica cyklu. Všetky signály sú však vyhodnotené pri prechode do poslednej polovice cyklu. Na selektoroch sa nevyskytujú žiadne nepotrebné prechody.

Výstup n4 selektora slova M2 v jednotke operandu je priamo pripojený na prvý vstup selektora slova M3, ktorý má štyri vstupy a k jeho druhému vstupu cez invertor INV. Výstup n2 je výstup exponenciálnej časti OP_e . Vstup B_{bias2} je pripojený na tretí vstup selektora slova M3.

Časť mantisy OP_m má päť vstupov v_{2i}, v_{1i}, a_1, a_2 a a_3 . Všetky vstupy sú pripojené ku každému vstupu selektora slova M4, ktorý má päť vstupov a výstup n8. Výstup n8 je pripojený na štvrtý vstup selektora slova M3 v exponenciálnej časti OP_e . Výstup n8 je tiež priamo pripojený na jeden z troch vstupov selektora slova M5, na druhý vstup selektora slova M5 cez negatívny posuvný register SH1, ktorý posunuje binárnu informáciu na svojom vstupe o štyri kroky v smere k nižšiemu významu, teda že delí binárnu informáciu šestnástimi a na tretí zo vstupov cez kladný posuvný register SH2, ktorý posúva binárnu informáciu na svojom vstupe o štyri kroky smerom k vyššiemu významu, to znamená, že násobí binárnu informáciu na svojom vstupe šestnástimi.

Výstup n7 selektora slova M5 je priamo pripojený na prvý zo šiestich vstupov selektora slova M6, cez negatívny posuvný register SH3, ktorý posúva binárnu informáciu na svojom vstupe o jeden krok smerom k nižšiemu významu na druhý vstup, cez druhý záporný posuvný register SH4, ktorý posúva binárnu informáciu na svojom vstupe o jeden krok smerom k nižšiemu významu na tretí vstup, cez kladný posuvný register SH5, ktorý posúva binárnu informáciu na svojom vstupe o jeden krok smerom k väčšiemu významu na štvrtý vstup a cez druhý posuvný register, ktorý posúva binárnu informáciu na svojom vstupe o jeden krok smerom k vyššiemu významu. Interný generátor konštantného slova 41 generuje konštantné slovo c_{word2} na svojom vstupe c_{210} a má svoj výstup pripojený na samostatný vstup selektora slova M6. Generátor

konštantného slova 41 môže vytvoriť skupinu dopredu definovaných kombinácií núl a jednotiek, napríklad všetky "0" alebo všetky "1" alebo slovo, ktoré udáva niektoré konkrétne informácie, riadené riadiacim signálom $c_{2_{10}}$ do generátora. Generátor si môže zapamätať iba obmedzený počet kombinácií slova, medzi ktorými vyberá riadiacim signálom $c_{2_{10}}$. Výstup selektora slova M6 je výstup n1 časti mantisy OP_m .

Časť mantisy OP_m používa nasledujúcu funkciu. Základná hodnota mantisy na n8 je vybraná selektorom slova M4, čo vedie na v_2, v_1, a_1, a_2 alebo a_3 . Použitie je možné bez zmeny alebo posunuté o 1, 2, 3, 4, 5, 6 bitov doľava (kladné posuny) alebo 1, 2, 3, 4, 5, 6 bitov doprava (záporné posuny) alebo môže byť nahradené slovom c_{word2} z generátora slova 41.

Exponenciálna časť OP_e používa nasledujúcu funkciu. Jej výstup môže byť buď neposunutá hodnota mantisy na n8, v_{2+1} , invertovaná v_{2+1} , v_{2-1} , invertovaná v_{2-1} , $B_{co-incr2}$, invertovaná $B_{co-incr2}$, $B_{co-decr2}$ alebo B_{bias2} .

V jednotke operandu r2 selektory slova M2 až M6 sú riadené individuálne riadiacim slovom na vstupe c_2 riadiaceho obvodu 27. Preto selektor slova M2 je riadený časťou riadiaceho slova c_{2_3} , selektor slova M3 časťou riadiaceho slova c_{2_2} , selektor slova M4 časťou riadiaceho slova c_{2_6} , selektor slova časťou riadiaceho slova c_{2_5} a selektor slova M6 časťou riadiaceho slova c_{2_4} . Tiež posuvné registre SH1 a SH2, pripojené na výstup n8 selektora slova M4, sú riadené časťou signálu c_{2_8} , posuvné registre SH3 až SH6 časťou signálu c_{2_7} , inkrementačný člen INC a dekrementačný člen DEC časťou signálu c_{2_9} , generátor konštantného slova 41, ktorý odovzdáva na výstup c_{word2} čiastočný signál $c_{2_{10}}$ a generátor 0/1 40 čiastočný signál $c_{2_{11}}$.

Individuálny selektor M1 bitovej skupiny je riadený kombinovaným multibitovým signálom, ktorý obsahuje čiastočné riadiace slovo c_{2_1} , výstupný kódový signál B_{cde2} z dekodéru presnosti PD2, výstupný signál najmenej významného bitu $B_{1sb\ adj2}$ a výstup signálu exponentu $B_{exp\ adj2}$, pričom posledné dva uvedené signály sú odvodené od procesných obvodov 42A a 42B z výstupov

B_{1sb2} a B_{exp2} , ktoré prichádzajú z dekodéru presnosti PD2. Signály budú popísané a zobrazené ďalej. Procesné obvody 42A a 42B sú riadené signálom $c2_5$, ktorý je časťou výstupu $c2$ z riadiaceho obvodu 27. Presný dekodér PD2 má znamienko exponentu a kódovú časť informácie na zbernici v_{2i} ako vstup spoločne s binárnym riadiacim signálom $c5_3$ z riadiaceho obvodu 27, ktorý udáva, či slovo na zbernici v_{2i} predstavuje hodnotu typu integer alebo s plávajúcou rádovou čiarkou. Riadiaci signál $c5_3$ je časťou výstupu $c5$ z riadiaceho obvodu ALU 27.

To isté platí pre presný dekodér PD1 pre jednotku operandu $r1$. Signály B_{1sb1} , B_{exp1} z dekodéru presnosti PD1 sú spracované v procesných obvodoch 43A a 43B, ktoré dávajú výstupy $B_{1sb\ adj1}$ a $B_{exp\ adj1}$. Procesné obvody 42A, 42B a 43A, 43B dávajú väčšinou rovnaký výstup z dekodéru, ku ktorému patrí. V niektorých prípadoch, keď z inkrementálneho člena INC alebo dekrementálneho člena DEC sa generuje prenos, výstupné signály budú posunuté, ako bude uvedené ďalej. Toto posunutie bude riadené časťou signálu $c2_5$ alebo $c1_5$ výstupu $c2$ alebo $c1$ z riadiacej jednotky 27 číslicovej ALU.

Presný dekodér PD2 (alebo PD1) dáva výstupy, ktoré sú všetky bezvýznamné, čo je napríklad indikované tým, že výstupy procesných obvodov 42A, 42B (43A, 43B) majú samé nuly, pokiaľ slovo v_{2i} predstavuje integer a výstupy druhu, ktorý je popísaný ďalej, ako B_{exp2} , B_{1sb2} a B_{cde2} (alebo B_{exp1} , B_{1sb1} a B_{cde2}), pokiaľ slovo v_{2i} (alebo v_{1i}) predstavuje plávajúcu rádovú čiarku.

Presný dekodér PD1 dáva výstupné signály $B_{co\ decr1}$, $B_{co-incr1}$, B_{BIAS1} , B_{exp1} , B_{1SB1} , B_{MSB1} a B_{CDE1} do jednotky operandu $r1$ a signál $B_{co\ max1}$, ktorý udáva, či kód znamená maximálnu dĺžku exponentu a signál $B_{co\ min1}$ udáva, či kód znamená minimálnu dĺžku exponentu do riadiaceho obvodu ALU 27. Presný dekodér PD2 odovzdáva na výstup signály $B_{co-decr2}$, $B_{co-incr2}$, B_{bias2} , B_{exp2} , B_{1sb2} , B_{msb2} a B_{cde2} do jednotky operandu $r2$ a signál $B_{co-max1}$ udáva, či kód predstavuje maximálnu dĺžku slova exponentu a signál $B_{co-min1}$ udáva, či kód predstavuje minimálnu dĺžku exponentu z riadiacej jednotky ALU 27. Výstupy B_{1sb} , B_{exp} z dekodéru presnosti PD1 alebo PD2 musia byť

posunutú rovnakým spôsobom ako je posunutá mantisa, aby sa nová presnosť zobrazila na výstupoch a v priebehu normálnych posunov, ktoré sú vykonávané riadením obvodov 42A a 42B alebo 43A a 43B pod riadením časti signálu $c2_5$ alebo $c1_5$, ako už bolo uvedené.

Na každý dekodér presnosti je privedený signál $c5_3$ na jeho vstup, čo indikuje, či slovo na vstupnej zbernici v_{1i} alebo v_{2i} predstavuje hodnotu integer alebo s plávajúcou rádovou čiarkou. Signál $c5_3$ je súčasťou signálu z výstupu $c5$ riadiaceho obvodu ALU 27, ktorý riadi obvody 23 až 30.

Výstupné signály z dekodéru presnosti budú neskôršie podrobne popísané.

Násobenie a delenie, jednotka operandu $r3$.

Tretia jednotka operandu $r3$, jednotka funkcie násobenia/delenia je iný typ ako jednotky $r1$ a $r2$ a je určená hlavne pre operácie násobenia a delenia. Jednotka $r3$ vykonáva posun a nastavenie, ktoré je nutné pre funkciu násobenia a delenia. Je riadené riadiacim slovom, ktoré je odovzdávané na výstupe $c3$ riadiacej jednotky ALU 27.

Jednotka násobenia/delenia $r3$ má niekoľko vstupov, z ktorých prvý dostáva informácie na zbernici v_{2i} a druhý informáciu na zbernici v_{3i} a tretí dostáva výstup z generátora konštantného slova 44, na ktorom je zvolená bitová kombinácia z niekoľkých uložených v generátore pomocou čiastočného riadiaceho signálu $c3_4$ výstupu $c3$ z obvodu 27.

Jednotka $r3$ sa používa na vyhodnotenie jedného argumentu počas zložitých číselných inštrukcií. Jej výstup je pripojený na zbernicu v_{3out} .

Na vstup v_{3i} je privedený prvý vstup selektora slova $M11$. Na vstup v_{3i} je privedený priamo druhý vstup selektora slova $M11$, na tretí vstup cez kladný jednobitový posuvný register $AH11$, ktorý je riadený signálom $c3_5$, na štvrtý vstup cez prvý záporný jednobitový register $SH12$ a na piaty vstup cez druhý záporný jednobitový register $SH13$ dva naposledy uvedené bitové posuvné registre, ktoré sú riadené signálmi $c3_3$, B_{msb1} , N_{1sb1} . Výstup generátora 44 je

pripojený na šiesty vstup selektora slova M11. Výstup selektora slova M11, ktorý má byť privedený na vstup druhého selektora slova M12, je zvolený časťou riadiaceho slova $c3_2$ riadiaceho slova $c3$ z riadiacej jednotky ALU 27.

M12 je zbernica, ktorá výstup selektora slova M11 dáva na výstup zbernice v_{3out} , ale ktorá je schopná riadenia do stavu vypnutia časťou riadiaceho slova $c3_1$.

Presný dekodér.

Princíp štruktúry dekodéru presnosti je uvedený na obr. 30. Je nutné poznamenať, že v uvedenej štruktúre môžu byť vykonané zmeny, napríklad na to, aby sa zvýšila rýchlosť výpočtu. Sú to iba konštrukčné vlastnosti a pre osobu skúsenú v odbore sú zrejmé a nebudú preto tu popísané. To znamená, že skutočný obvod môže mať úplne inú konfiguráciu v porovnaní s obr. 30, ale ktoré budú mať základnú funkciu, ktorá vyplýva z obr. 30.

Znamienko exponentu a kódová časť vstupného signálu v_{ji} , kde j je buď 1 alebo 2, sú privedené na záznamový obvod 50, ktorý v tomto zapojení na svojom výstupe dáva jeden z variantov kódu vrátane vedúceho znamienka exponentu, ktoré je uvedené v tabuľke hore, napríklad vľavo, nech je na vstup privedený akýkoľvek variant. Alebo obidva varianty môžu byť privedené na výstup záznamového obvodu 50, ale vždy v rovnakom poradí, napríklad ľavý variant v tabuľke hore na priamkach, ktoré sú mu priradené a pravý na priamkach, ktoré sú mu priradené. Výstup je privedený na obvod komparátora bitovej postupnosti 51, ktorý sleduje kombináciu bitov na jeho vstupe a dáva číslícový signál B_{msb} , ktorý udáva bitovú dĺžku kódu. Vytvára tiež signál B_{cde} .

Obvody 52 až 57 komparátora šesťbitovej postupnosti sú tiež pripojené na výstup prekódovacieho obvodu 50, každý je prispôsobený na reagovanie na svoje konkrétne kódovanie, pri ktorom dáva "1" na výstupe, ak bitový vzorec na výstupnom vodiči z prekódovacieho obvodu je rovnaký, ako je prispôsobený reagovať, aby na výstup odovzdal "1" namiesto svojho normálneho výstupu "0". Iba jeden z obvodov 52 až 57 v určitom momente dáva "1", teda signál true, ostatné dávajú na výstupe "0", teda signál false. Prvý a

posledný bit v signále B_{1sb} je vždy "0". Preto dva vodiče 59 a 59 sú pripojené priamo na vodič, ktorý má bit "0".

Vodič 58, výstupy z obvodov 52 až 57 a vodič 59 tvoria zbernicu B_{1sb} , ktorá má jeden bit na skupinu presnosti. Zbernica B_{bias} má štyri bity na skupinu presnosti. Každá skupina obsahuje tri bity "0" a bit signálu B_{1sb} pre túto konkrétnu skupinu. Preto tri z vodičov z každej skupiny sú pripojené na "0" a štyri na príslušný vodič výstupu B_{1sb} . Výstup B_{co_min} je pripojený na výstup obvodu 52 a výstup B_{co_max} je pripojený na výstup obvodu 57.

Bitový vzorec výstupu obsahuje "0":s na najmenej významnej pozícii až do pozície, kde výstup B_{1sb} má "1" a potom sú všetky bity "1":s. Najmenej významný bit na vodiči 59 bude vždy "0", ale z poslednej najmenej významnej pozície plyní možnosť pre bit "1". Preto hradlo OR má jeden vstup pripojený na výstup obvodu 57 a ostatné na výstup obvodu 56. Výstup obvodu 57 je pripojený na ďalší vodič s najmenším významom signálu B_{exp} . Výstup hradla OR 60 je pripojený na druhý, ďalší vodič s najmenším významom výstupu B_{exp} . Pokiaľ teda výstup obvodu 57 by mal byť "1", potom tento výstup bude mať tiež bit "1" na výstupe hradla OR 60. Výstup z hradla OR 60 bude "0" iba keď výstupy z obidvoch obvodov 57 a 56 sú "0". Výstup hradla OR 60 je pripojený na prvý vstup ďalšieho hradla OR 61, ktoré má svoj vstup pripojený na výstup obvodu 55. Výstup hradla OR 61 je pripojený na tretí vodič s najmenším významom výstupu B_{exp} a na prvý vstup hradla OR 62, ktoré má svoj vstup pripojený na výstup obvodu 54, atď. Tento druh kaskádového zapojenia hradiel OR dáva "1":s vo všetkých pozíciach, ktoré majú rovnaký alebo vyšší význam ako pozícia, ktorá dáva "1" na výstupe B_{1sb} .

Znamienko, znamienko exponentu a kódová časť slova $B_{co-decr}$ má svoj prvý vodič pripojený na jednotku, ktorá dáva "1", svoj druhý vodič na invertor 161 má pripojený k šiestemu vodiču B_{1sb} , tretí vodič na hradlo OR 162, ktoré má svoje vstupy pripojené na piaty a šiesty B_{1sb} a svoj štvrtý a piaty vodič pripojený priamo na druhý a tretí vodič B_{1sb} . Tieto vodiče pre znamienko, znamienko

exponentu a kódovú časť slova $B_{co-decr}$ sú pripojené na vstup bitového doplnujúceho obvodu 66, ktorý pričíta "0":s k prichádzajúcemu slovu, aby sa na jeho výstupe objavilo 32-bitové slovo.

Znamienko, znamienko exponentu a kódová časť slova $B_{co-incr}$ majú svoj prvý a druhý vodič pripojený na jednotku, ktorá zaistuje "1", tretí na invertor 163, pripojený na siedmy vodič B_{1sb} , štvrtý na hradlo OR 164, ktoré má svoje vstupy pripojené na tretí a štvrtý vodič B_{1sb} a šiesty vodič pripojený priamo na hradlo OR 165, ktoré má svoje vstupy pripojené na tretí a piaty vodič B_{1sb} . Tieto vodiče pre znamienko, znamienko exponentu a kódovú časť slova $B_{co-incr}$ sú pripojené na vstup bitového doplnkového obvodu 67, ktorý pričíta "0":s k prichádzajúcemu slovu, aby sa na jeho výstupe objavilo 32-bitové slovo.

Sčítačka.

Zapojenie sčítačky ako jedného z obvodov je uvedené na obr. 28 až 30 (z obr. 28 je uvedené na obr. 31A až 31C). Sčítačka má dva argumenty a a b . Argument a je odovzdávaný na zbernicu v_{1i} a argument b na zbernicu v_{2i} . Tieto argumenty sú kódované buď ako hodnoty s plávajúcou rádovou čiarkou alebo ako integer. Signál $c5_1$, ktorý je "0" pre hodnoty integer a "1" pre "0" pre hodnoty s plávajúcou rádovou čiarkou, z výstupu $c5$ obvodu 27 na obr. 28 a zbernice B_{1sb1} a B_{1sb2} , ktoré majú všetky bity "0" pre hodnoty integer a jeden bit "1" pre hodnoty s plávajúcou rádovou čiarkou, indikujú túto skutočnosť. Ako je uvedené na obr. 31A, sčítačka je binárna sčítačka, postavená z ôsmich kaskádne zapojených sekcií $G0, G1, \dots, Gi, \dots, G7$, to znamená jednu sekciu pre každú skupinu presností.

Ako je zrejmé z vyššie uvedeného popisu operácie sčítania, je vhodné porovnať argumenty a a b v prípade, že a a b sú hodnoty s plávajúcou rádovou čiarkou ešte predtým, ako môžu byť sčítané. To znamená, že B_{1sb1} a B_{1sb2} musia byť rovnaké. B_{1sb1} je privedené ako signál B_{1sb} aby riadil sčítačku. Signál B_{1sb} je posunutý o jeden krok v smere k mantise, keď je pripojený na sčítačku, inkrementačnému alebo dekrementačnému členu atď, pretože

by potom mohol ovplyvniť postup prenosu (zablokovať ho) v skupine najvýznamnejšej mantisy. Preto najmenej významný bit v signále B_{1sb} sa v týchto prípadoch nepoužíva.

Argumenty a a b , kde každý obsahuje 32 bitov, sú rozdelené na osem skupín, pričom každá z nich obsahuje 4 bity. Každá skupina a a b po štyroch bitoch je privedená ako samostatná sekcia G_i , kde i je číslo integer medzi 0 a 7. Každý vodič zbernice b_{1sb} je pripojený na samostatnú sekciu G_i okrem G_0 . Vodič c_{in} k sekcii G_7 indikuje, či sčítačka bude vykonávať sčítanie alebo odčítanie. c_{in} je "0" pre sčítanie a "1" pre odčítanie. Každá sekcia odovzdáva signál prenosu, ktorý je privedený na ďalšiu sekciu a je privedený so signálom c_{ini} z ďalšej predchádzajúcej sekcie, ako bude podrobnejšie popísané ďalej.

Sčítanie argumentov a a b sa vykonáva v každej sekcii G_0 až G_7 individuálne, pričom pokiaľ je nutné, odovzdáva do ďalšej sekcie signály prenosu v poradí G_7 až G_0 . Pre hodnotu s plávajúcou rádovou čiarkou je odovzdávaný prenos mantisy von zo zbernice c_m , na ktorú sa uskutočňuje prenos mantisy von, pokiaľ je to možné. Táto zbernica spoločne s vodičmi c_s , ktoré majú znamienkový bit a c_{se} , ktoré majú znamienkový bit exponentu, tvorí vstup gr_{a1} (alebo gr_{a2} alebo gr_{a3}) pre obvod 27 na obr. 28. Obvod 27 dáva túto informáciu do riadiacej jednotky 6 na zbernicu A_{STATE} .

Signál $c5_1$ udáva, či argumenty a a b sú hodnoty typu integer alebo s plávajúcou rádovou čiarkou, signál $c5_2$ udáva, či kód je normálny alebo pozmenený pre argument a a signál $c5_3$ udáva, či kód je normálny alebo pozmenený pre argument b . Tieto signály sú privedené na sekciu G_0 , ktorá odpovedá najvýznamnejšej skupine presnosti vrátane najvýznamnejšieho bitového rezu (znamienkový rez). Toto je závislé na formáte operandov a a b (či sú obidva integer alebo s plávajúcou rádovou čiarkou).

Trojstavové hradlo 70 nastavuje zbernicu c_m na definovanú hodnotu, napríklad všetky "0", keď operandy sú typu integer, pretože sa na zbernici neaktivuje žiadny prenos mantisy. Trojstavový driver 70 je riadený signálom $c5_3$.

Zapojenie konfigurácie sekcie G_1 je uvedené na obr. 31B. Je

"rozrezané" na štyri sčítacie rezy AS1 až AS4. Obsahuje tiež generátor prenosu pre sekciu, ktorá vyhodnocuje prenos $c_{out\ i}$ pre ďalšiu sekciu. Každá bitová skupina a_i a b_i pre danú sekciu G_i má príslušné bity a_{ik} a b_{ik} pripojené na každý samostatný rez sčítačky ASk, kde k je celé číslo medzi 1 až 4. Každý rez sčítačky odovzdáva jeden bit sum_{ik} výstupu súčtu.

Ako bude popísané podrobnejšie pri popise obr. 31C, generátor prenosu má bit pre generovanie Gen, ktorý indikuje, že súčet sekcie je väčší alebo rovný 16, bit postupu Pr, ktorý indikuje, že súčet sekcie je väčší alebo rovný 15, bit prenosu z predchádzajúcej sekcie $c_{in\ i}$, bit prenosu c_{in-i4} z rezu sčítačky AS4 a bit $B_{1sb-i+1}$. Bit $B_{1sb-i+1}$ je privedený cez invertor 75 na vstup invertora trojstavového hradla a priamo na riadiaci vstup trojstavového hradla 76. Bit c_{in-i4} je pripojený na vstup druhého invertora trojstavového hradla 76. Preto keď bit $B_{1sb-i+1}$ je "0", potom trojstavové hradlo 76 je nastavené na vysielanie bitu $B_{1sb-i11}$. Keď však bit $B_{1sb-i11}$ je "1", potom trojstavové hradlo 76 je nastavené na vysielanie bitu c_{in-i4} , ktorý, keď je V "1" udáva, že mantisa má prenosový bit.

Je iba nutné zaistiť výstup prenosu mantisy do skupiny presnosti vedľa exponenciálnej časti argumentu a iba táto skupina má bit $B_{1sb-i+1}$ v "1".

Postup prenosu zo sekcie na sekciu je blokovaný pre skupinu presnosti vedľa exponenciálnej časti. Preto bit $B_{1sb-i+1}$ je pripojený na invertovaný vstup hradla AND 77, ktoré má invertovaný signál Gen na svojom druhom invertovanom vstupe a vysielá takto signál Gen iba keď $B_{1sb-i+1}$ je "0". Bit $B_{1sb-i+1}$ je tiež pripojený na invertovaný vstup hradla AND 78, ktoré má invertovaný signál Pr na druhom invertovanom vstupe a bit $c_{in\ i}$ na treťom invertovanom vstupe a tak vysielá signál pr iba vtedy, keď $B_{1sb-i+1}$ je "0" a $c_{in\ i}$ je "0". Hradlo OR 79 má svoje vstupy pripojené na výstupy hradiel AND 77 a 78 a má na výstupe "1", keď jedno z hradiel AND má na výstupe "1".

Zapojenie každého rezu sčítačky ASk je uvedené na obr. 31C. Vstupy a_{ik} a b_{ik} sú privedené na vstup každého hradla EXCLUSIVE

OR 80, ktoré dáva "1" ak vstupy sú navzájom odlišné, ináč je na výstupe "0". Výstup EXCLUSIVE OR hradla 80 je privedený na prvý vstup druhého hradla EXCLUSIVE OR 81 a bit c_{IN_ik} na druhý vstup. Výstup hradla EXCLUSIVE OR 81 je "1" iba keď jeho vstupy sú navzájom rôzne a má na výstupe sum_{ik} , ktorý zahrňuje prenosový bit z ďalšieho rezu sčítačky alebo ak rez sčítačky je prvý v sekcii, prenosový bit z ďalšej predchádzajúcej sekcie.

Bity a_{ik} a b_{ik} sú privedené na vstupy hradla OR 82, ktoré má invertovaný výstup, ktorý je pripojený na prvý invertovaný vstup každého z troch hradiel AND 82, 84, 85. Invertovaný bit $c_{in_ik}^*$ je privedený na druhý invertovaný vstup hradla AND 83, ktorý vysiela bit c_{in-ik} , pokiaľ je niektorý bit a_{ik} a b_{ik} v "1". Invertovaný bit $P_{in_ik}^*$ z ďalšieho predchádzajúceho rezu sčítačky je privedený na druhý invertovaný vstup hradla AND 84, ktoré vytvára bit P_{out-ik} . Tento bit je P_{in-ik} ak niektorý z bitov a_{ik} a b_{ik} je "1". Invertovaný bit G_{in-ik}^* je privedený na druhý invertovaný vstup hradla AND 85, ktorý odovzdáva bit G_{in_ik} pokiaľ niektorý z bitov a_{ik} a b_{ik} je "1". Bit P_{in_ik} a bit G_{in_ik} sú "1" respektíve "0" pre každý rez sčítačky AS1 v každej sekcii presnosti, ako je zrejmé z obr. 31B.

Bity Pr a Gen sú nutné k tomu, aby sa realizoval zvlnený prenos, teda aby bola sčítačka rýchlejšia. Vzhľadom ku skutočnosti, že na vstupoch reťazca prenosu sú diskkrétne hodnoty, všetky skupiny presnosti sú počítané paralelne to znamená súčasne pre všetky presné skupiny. Posledný prenos zo sčítania bude mať oneskorenie iba na 4+8 hradlách v porovnaní s oneskorením na 32 hradlách bez zvlneného prenosu.

Bity a_{ik} a b_{ik} sú tiež privedené na vstupy hradla AND 80, ktoré má invertovaný výstup, ktorý je pripojený na prvý invertovaný vstup obidvoch hradiel OR 86, 87. Výstup hradla AND 83 je pripojený na druhý vstup hradla OR 86, na ktorého výstupe je signál c_{out-ik} . Výstup hradla 85 je pripojený na druhý vstup hradla OR 87, ktoré vytvára výstup G_{out_ik} .

Aby sa zaistil veľmi rýchly reťazec prenosov, je možné mať v ňom iba jedno oneskorenie, pokiaľ bitový reťazec invertuje

polaritu zvlnenia a bity šírenia sú invertované. V takom prípade má každá druhá skupina presnosti invertované vstupy (neuvedené).

Ako je zrejme z ľavej strany obr. 31A, najvýznamnejšia sekcia presnosti G_0 je čiastočne odlišná od ostatných sekcií. Nepoužíva vstupný argument B_{1sb_0} . Predpokladá sa, že vo vnútri skupiny je "0". Namiesto toho má vstup pre prenos mantisy c_{in_m} . Je použitý pre generovanie vstupu prenosu z exponentu. Normálny proces je generovaný ako v normálnych skupinách okrem toho, že sa predpokladá, že B_{1sb_0} je "0", teda false.

Jednabitové signály $c5_1, c5_2, c5_3$ môžu mať nasledujúce bitové vzorce: $(c5_1, c5_2, c5_3) = (false, false, false)$ znamená typ integer, $(c5_1, c5_2, c5_3) = (false, false, true)$ znamená, že a i b majú normálny kód, $(c5_1, c5_2, c5_3) = (true, false, false)$ znamená, že a má zmenený a b normálny kód, $(c5_1, c5_2, c5_3) = (true, true, true)$ znamená, že a má normálny a b zmenený kód.

Inkrementačný člen.

Konfigurácia inkrementačného člena je veľmi podobná konfigurácii sčítačky. Obr. 32 ukazuje zapojenie sekcie presnosti v inkrementačnom člene. Inkrementačný člen má iba jeden vstup a (nie je uvedený). Bitový reťazec inkrementačného člena AS_1 až AS_4 sa líši od bitového reťazca sčítačky v tom, že sa neprenáša z jedného rezu na druhý generovaný bit Gen . Invertovaný vstup Pr z rezu AS_4 a invertovaný prenos c_{in}^* z ďalšej sekcie presnosti sú privedené na obidva invertované vstupy hradla AND 90 s dvomi invertovanými vstupmi. Prenos v c_{in} do prvej sekcie G_7 na linke, ktorá je "1" hlási, že má byť vykonaná inkrementácia. Výstup hradla AND 90 a bit B_{1sb_i+1} sú privedené na vstup hradla OR 92, ktoré generuje výstup prenosu c_{out} do ďalšej sekcie. Výstup prenosu inkrementačného člena prebieha rovnakým spôsobom ako u sčítačky. Reťazec zvlneného prenosu c_{in}/c_{out} indikuje, či súčet je väčší alebo rovný dvom. Reťazec zvlneného prenosu Pe indikuje, či súčet je väčší alebo rovný jednej.

Ako sčítačka tak i inkrementačný člen má zbernicu výstupu prenosu mantisy c_m , ktorá je riadená signálom B_{1sb_i+1} . Riadiaci bit $c1g$ alebo $c2g$ udáva, či reprezentácia hodnoty je typu integer

alebo s plávajúcou rádovou čiarkou. Riadiace bity $c1g$ a $c2g$ riadia trojstavové hradlo, ktoré nastavuje zbernicu výstupu prenosu mantisy na definovanú hodnotu, keď nie je používaná (neuvedené).

Dekrementačný člen má rovnakú konfiguráciu ako inkrementačný člen ale dekrementačný člen má invertovaný vstup a i_n na svojich rezoch dekrementačného člena, ktorý odpovedá rezom inkrementačného člena AS1 až AS4.

Selektory.

Selektory v jednotke operandu sú obecné zostavené z množstva vysielacích hradiel. Selektor má vždy definovaný výstup. Má zoznam riadiacich slov a hodnôt na vstupoch ako argumenty. Všetky okrem jedného sú vypnuté. Riadenie každého vstupu je komplementárny signál. Celý selektor je riadený zoznamom takých prvkov. Sú riadené v tomto odlišnom stave, aby sa prevzal vstup z predpísaného portu. Zapojenie základného bitového rezu selektora je uvedené na obr. 33. Jeden z bitov každého vstupného portu do selektora S_{ina} , S_{inb} a S_{ind} je zapojený na vstup riadeného prepínača S_a , S_b a S_d . Pár riadiacich vodičov S_{ca} , S_{cb} , S_{cc} a S_{cd} je určený pre každý prepínač. Riadiaci pár, ktorý zaisťuje prenos vstupu na jeho príslušný prepínač, má informáciu "1,0", ostatné majú informáciu "0,1".

Je nutné poznamenať, že väčšina obvodov v číslicovej ALU je postavená podľa princípov, popísaných vyššie vo vzťahu ku sčítačke a že sa preto predpokladá, že skúsená osoba môže ľahko vykonštruovať zvyšok obvodov podľa tohto princípu. Preto sa nepokladalo za nutné popisovať každý obvod podrobne.

Popis riadiaceho obvodu číslicovej ALU 27

Riadiaci obvod číslicovej ALU 27, uvedený na obr. 28, je boolovské hradlové pole pomerne zložitej konfigurácie. Jeho zapojenie nemôže byť uvedené, pretože jeho zapojenie je vypočítavané na počítači, ktorý je vybavený všetkými algoritmami, potrebnými ku konfigurácii riadiacich signálov vo vzťahu k informatívnym signálom. Boolovské hradlové pole je urobené priamo ako maskované vzorce na čipe z počítača, prispôso-

beného konfigurácii maskovaných vzorcov na čipe pre celú číslicovú ALU podľa vynálezu. Preto je nemožné tento obvod uviesť podrobnejšie. Zoznam vzťahov medzi vstupmi a výstupmi z riadiacej jednotky 27 je uvedený v prílohe 2 pre prípad násobenia čísiel typu integer.

Príloha 2 ukazuje vzťah medzi vstupmi do a výstupmi z riadiaceho obvodu ALU 27, keď sa vykonáva násobenie čísiel integer. Príloha je rozdelená na dva odlišné "riadiace prípady". Každý riadiaci prípad odpovedá jednému výstupu, teda jednej makroinštrukcii.

Výstupy riadiaceho obvodu ALU 27, ktoré sú uvedené pod hlavičkou "vodiče mikroinštrukcie", sú úrovne signálu na zbernici C1-C5. Zbernice C1, C2 a C3 sú rozdelené na menšie zbernice C₁₁, C₁₂, C₁₃ atď. Podľa tohto vstupy, ktoré sú uvedené pod hlavičkou "riadiaci prípad 1" ako "stav, snímací vodič - podmienka 4" majú za následok výstupy pod hlavičkou "vodiče mikroinštrukcie" nižšie. Vstupy, ktoré sú uvedené vyššie ako podmienka 1, 2 a 3 majú za následok rovnaké výstupy ako za podmienky 4.

Každá podmienka je definovaná určitou kombináciou úrovní signálu (0/1) na niektorých stavových vodičoch, zatiaľ čo stavové vodiče môžu mať nedefinované úrovne (X = ľubovoľný stav).

Všetky vstupné signály uvedené v prílohe sú tiež vyznačené na obr.28, okrem signálov co_limit1 a co_limit2.

Na obr.28 je signál co_limit1 rozdelený na dve zložky B_{co_max1} a B_{co_min1}.

Signál co_limit2 je tiež rozdelený na dve zložky B_{co_max2} a B_{co_min2}.

Výstup A_{STATE} je obyčajnou kópiou všetkých vstupov do obvodu 27. Výstup A_{STATE} je pripojený na riadiacu jednotku 6.

Hlavnými registrami v jadrovej bunke v procesore prvého rádu alebo niektorými z vnútorných jednotiek číslicovej ALU môže byť odovzdaná hodnota ako inštrukcia operand/mul_quotient alebo polynóm. Informácia týkajúca sa druhu reprezentácie vo zvyšku reťazca na zbernici môže byť daná informáciou na informačnej zbernici alebo môže byť daná v kódovom poli.

Port.

Redukčný procesor môže byť vybavený jedným alebo niekoľkými portami pre vstup/výstup dát do a z procesoru. Každý port je jednotka, ktorá zaisťuje komunikáciu s externým prostredím procesoru. Port je najlepšie pripojený na transformačný interfejs 7 v objektovom ukladaní 1 (viď obr.1). Kaď je však trochu modifikovaný, môže byť miesto toho pripojený na prostriedok pre prenos dát 4. Na port je možné pozerat ako na predĺženie objektového ukladania, pretože obsahuje štyri pamäťové bunky, ktoré sú podobné vo funkcii a kompaktilné s bunkami objektového ukladania 10 (viď obr.2). Port vykonáva vstupné/výstupné operácie cez unifikáciu.

Interné chovanie.

Objektové ukladanie vytvára vnútorné chovanie procesoru, samostatné pre každý port. Chovanie je sémantický význam časovej štruktúry, ktorá môže byť prezentovaná spínačovou štruktúrou. Časová štruktúra môže byť pokladaná za stav procesoru, zloženého nekonečnou časovou postupnosťou hodnôt.

Externé chovanie.

Vstupné signály do portov môžu byť odovzdávané snímačmi alebo zariadeniami mimo procesor, teda v prostredí reálneho sveta. Port transformuje vstupujúce signály na číslicový tvar, prijateľný pre formát objektového ukladania, teda na reprezentáciu spínača. Táto reprezentácia spínača je časová štruktúra, teda chovanie, porovnateľné s chovaním procesora. Preto z hľadiska procesora táto časová štruktúra môže byť považovaná za objektové chovanie.

Vstup a výstup cez port sa vykonáva unifikáciou procesorov, ktoré prebiehajú na každej strane, teda interné chovanie v procesore a externé chovanie v jeho okolí. Ako vnútorné tak i vonkajšie chovanie sú modelované rovnakým spôsobom, ktorý dáva elegantný spôsob operácie a robí tak procesor vhodným pre vykonávanie operácií v reálnom čase.

Príklad takého externého chovania, teda časovej postupnosti

je uvedený na obr.34. Postupnosť vzorkovaných signálov sa môže meniť s časom, to znamená, že vzorkovacie periódy sa môžu líšiť, ako je zrejmé z obr.34. Postupnosť signálov je tu daná zoznamom párov prvkov. Každý pár obsahuje dobu trvania a signálovú hodnotu počas tejto doby. Signálové hodnoty sú označené q_i a doby trvania sú značené t_i , kde i je číslo medzi 1 a 6.

Interface.

Interface by mal mať nasledujúce vlastnosti:

1. Signál je meraný v určitých momentoch. Meraný signál sa považuje za konštantný až do ďalšieho merania.
2. Časový moment by mal byť daný buď pevným cyklom, ktorý je určený portom alebo pevným cyklom, ktorý je určený programom počítača alebo určitými intervalmi, ktoré sú určené programom alebo externým hodinovým signálom.
3. Meraný signál je odovzdaný do počítača v digitálnom tvare.
4. Digitalizovaný signál je buď boolovský alebo je kódovaný do digitálneho tvaru, ktorý môže určovať hodnotu typu integer alebo s plávajúcou rádovou čiarkou.

Procesor podľa vynálezu nevkladá alebo neodovzdáva na výstup hodnoty ako konvenčný procesor. V procesore podľa vynálezu je port ako interfejs medzi programom procesoru a okolím procesoru. Program je uložený v aktívnom objektovom ukladaní a vytvára vnútorné chovanie.

Vykonávanie programu je založené na tom, že program v procesore a okolí pracujú s rovnakým priebehom reálneho času dejov v interfejsse.

Vstup a výstup z procesora je vykonávaný pomocou unifikácie. To znamená, že doba trvania a hodnota vo vnútri každého intervalu sú rovnaké na oboch stranách portu.

Program môže špecifikovať konkrétnu dobu trvania alebo môže ponechať dobu trvania nešpecifikovanou, keď sa použije konkrétny symbol, napríklad \$, ktorý označuje všetky možné hodnoty.

Správny program by mal mať po operácii unifikácia ponechané iba jedno z možných chovaní a toto chovanie sa musí zhodovať s chovaním na danom porte. Pokiaľ by žiadne z možných vnútorných

chovaní neodpovedalo prostrediu reálneho sveta na danom porte, operácia unifikácia odovzdá ako výsledok nothing (nič), čo je považované za chybu programovania, pretože nothing znamená rozpor. Keby nám zostalo viac alternatívnych chovaní po operácii unifikácia, je to tiež považované za chybu programovania.

Vstup a výstup.

Interval je popísaný v programe v tvare during (počas) t v , kde t je doba trvania a v je signál hodnoty. Keď interné chovanie špecifikuje hodnotu signálu, ktorá sa tiež nazýva hodnota signálu, táto špecifická hodnota signálu je odovzdaná na výstup do prostredia. Pokiaľ interné chovanie ponechá hodnotu signálu nešpecifikovanú, z prostredia sa prevezme zo vstupu špecifická hodnota signálu.

Keď interné chovanie špecifikuje konkrétnu dobu trvania v intervale, potom okolie pripúšťa ľubovoľnú dĺžku trvania. Pokiaľ interné chovanie ponechá dobu trvania nešpecifikovanú, okolie dobu trvania určí.

Zapojenie portu.

Zapojenie portu pre vstup a/alebo výstup je uvedené na obr. 35. Port je pripojený na transformačný interfejs 7 s usporiadaním zbernice DU. Port je pripojený na centrálnu jednotku 6 procesoru prvého rádu, ku ktorému je pripojený a je nim riadený. Port obsahuje identifikačný register P_{ID} , ktorý označuje port. Identifikátor uložený v registri P_{ID} môže byť použitý ako identifikátor v objektovom ukladaní 1, teda že môže byť použitý na spojenie portu s bunkovým spínačom v objektovom ukladaní. Tento identifikátor sa používa iba interne vo vnútri procesoru. Port tiež zahrňuje register identifikátora CE_{ID} , ktorý môže byť použitý podobným spôsobom ako je register P_{ID} . Port tiež obsahuje veľa pamäťových buniek CE_{DU} , CE_{LAST} , CE_{NEXT} , z ktorých sa každá chová ako pamäťová bunka v objektovom ukladaní.

Každá pamäťová bunka si môže pamätať spínač, ktorý má minimálne tri pamäťové polia pre uloženie minimálne troch spínacích prvkov, napríklad during, dobu trvania a hodnotu signálu.

Minimálne dva z uvedených spínacích prvkov môžu byť uložené

v pamäťových poliach druhu hodnota/označenie. Výraz during sa môže vyskytnúť v poli TYPE. Pretože port vždy vykonáva operáciu during, toto pole TYPE môže byť vynechané.

V zapojení na obr.35 je during v tvare typu apply ('g'), umiestnený v poli TYPE spolu s dopredu definovaným názvom funkcie (kód during). Názov funkcie je uložený v prvom poli hodnota/označenie pamätevej bunky portu ako pre všetky názvy funkcií (ako je + - * /), ktoré patria typu apply. Kód during môže byť identifikátor, ktorý označuje definíciu funkcie, ktorá je uložená v objektovom ukladaní.

Identifikátor id môže byť tiež uložený v poli hodnota/označenie. Tento identifikátor spojuje štruktúru portu so štruktúrou during v objektovom ukladaní.

Pokiaľ pole, ktoré obsahuje identifikátor, nie je potrebné, potom sú potrebné iba tri polia hodnota/označenie. Môže sa ale tiež vyskytnúť štvrté pole s označením unused, aby registre portu boli kompaktilné s ostatnými pamäťovými bunkami v procesore. Je ale tiež možné pripojiť samostatné zariadenie pre VALUE IN/OUT ku štvrtému registru hodnota/označenie. Tým sa vytvoria porty pre skupiny hodnôt, teda pre trvanie času a dva druhy signálových veličín.

Každá pamäťová bunka tak môže mať pole atribútov. Pamäťové bunky CE_{DU} , CE_{LAST} , CE_{NEXT} môžu byť prepojené zo zbernicami V_{e1} , V_{i1} , V_{t1} , V_{v1} , V_{E1} , V_{T2} , V_{v2} , V_{E2} , pričom každá zbernica má napríklad 38 vodičov a je pripojená spôsobom bit po bitu k bitovým bunkám každého registra. Index I označuje zbernicu, ktorá je pripojiteľná ku kódovým poliam registrov during, index T označuje zbernicu, ktorá je pripojiteľná k poliam registrov hodnota/označenie doby trvania, index V označuje zbernicu, ktorá je pripojiteľná k poliam registrov signálu veličiny hodnota/označenie, index E označuje zbernicu, ktorá je pripojiteľná k poliam registrov samostatnej hodnoty/označenie. Index 2 označuje zbernicu, ktorá je pripojiteľná k okoliu a index 1 označuje zbernicu, ktorá je pripojená na transformačný interfejs 7.

Pamäťové bunky v uvedenom zapojení portov majú uložený ďalší

predchádzajúci spínač, aktuálny spínač, ďalší spínač a identifikátor budúcich spínačov. Pamäťové bunky môžu mať predpísané miesta a potom celý ich obsah môže byť medzi nimi prenesený. Vhodná štruktúra pre každú registrovú bunku v registroch môže mať rovnakú štruktúru v jadrovej bunke 2 (viď obr. 10).

Je však možné usporiadať tri prvé uvedené pamäťové bunky tak, že môžu vymeniť miesto navzájom zmenou svojich názvov, čo je urobené riadením z centrálnej riadiacej jednotky 6.

Presné zapojenie medzi pamäťovými bunkami nie je uvedené podrobne, pretože sa predpokladá, že skúsená osoba si môže ľahko odvodiť to isté pomocou popísaných princípov vzhľadom k popisu jadrovej bunky. Riadenie z riadiacej jednotky 6 je schématicky uvedené napravo na schématicky uvedenom zapojení medzi pamäťovými bunkami, ale je normálne načrtnuté ku každému spínaču v registrových bunkách ako v registroch jadrovej bunky.

Port môže mať rôzny počet pamäťových buniek, napríklad jednu pamäťovú bunku.

Čítač času a komparátor času.

Port obsahuje čítač času T_{COUNT} . Je riadený hodinovým signálom $CLOCK_p$ z centrálnej riadiacej jednotky 6 a meria čas. Môže byť resetovaná buď externým resetovacím signálom alebo resetovacím signálom z riadiacej jednotky 6. Čítač času T_{COUNT} je pripojený na prvý selektor SEL_1 , ktorý je riadený z centrálnej riadiacej jednotky 6, aby pripojil výstup z čítača času T_{COUNT} buď priamo na zbernicu V_{T2} , ktorú je možné pripojiť na polia časového ukladania v pamäťových bunkách alebo na prvý vstup časového komparátora T_{COMP} . Druhý vstup časového komparátora T_{COMP} je pripojený na zbernicu V_{T2} . Výstup časového komparátora je výstup externého času. Výstup časového komparátora je tiež prenesený na riadiacu jednotku 6.

Port tiež obsahuje vstup externej hodnoty a špičku výstupného portu $VALUE\ IN/OUT$. Je pripojený na prevodník $CONV_{OUT}$ pre výstupný signál, napríklad číslicovo/analógový prevodník. Je tiež pripojený na prevodník $CONV_{IN}$ pre vstupný signál, napríklad analógovo/číslícový prevodník, ktorý pracuje so vzorkovaním

vstupného signálu. Časy vzorkovania môžu byť dané výstupom komparátora T_{COMP} alebo resetovacími signálmi do čítača času T_{COUNT} . Selektor SEL_2 , pripojený na zbernicu V_{V2} , ktorá je riadene pripojiteľná k registrom signálovej veličiny pamäťových buniek CE_{DU} , CE_{LAST} , CE_{NEXT} , volí z centrálnej riadiacej jednotky $\underline{6}$, pokiaľ signál veličiny bude privedený z alebo na port. Riadiaca jednotka $\underline{6}$ najskôr prečíta obsah registrov v pamäťovej bunke, ktoré majú byť pripojené k prostrediu, aby sa videlo, či obsahuje konkrétny symbol \$. Pokiaľ áno, prečíta sa vstup z prostredia. Pokiaľ sa miesto toho zistí špecifická hodnota, urobí sa výstup do prostredia. Potom riadiaca jednotka $\underline{6}$ bude riadiť selektor portu, ktorý je príslušne pripojený k registru.

Je tiež možné vytvoriť zložitejší port. Napríklad veličina signálu pre VALUE IN/OUT môžu byť prevedené podľa vzorca, ktorý je uložený v signálovom prevodníku SC, pripojeného medzi selektor SEL_2 a zbernicu V_{V2} . Funkcia prevodu môže napríklad byť niekoľko integrálov, každý pre inú triedu hodnoty veličiny signálu.

Informácia, ktorá má byť privedená von, je v pamäťovej bunke CE_{DU} . Preto bunka bude mať informáciu during t v, kde t je doba trvania a v je veličina signálu. Veličina signálu v tomto registri je známa a bude privedená von. Veličina signálu je načítaná z registra veličina signálu v pamäťovej bunke a je privedená von cez selektor SEL_2 , ktorý pracuje ako riadiaci stupeň a prevodník $CONV_{OUT}$.

Vstup veličiny.

Informácia v pamäťovej bunke CE_{DU} je during t \$, kde t je doba trvania a \$ predstavuje akúkoľvek zo všetkých možných veličín signálu, teda že tento výraz sa zjednotí s každou veličinou signálu. Veličina signálu v tomto registri je teda neznáma a register je pripravený na príjem veličiny signálu. Veličina signálu je načítaná z veličiny signálu na portu vstup/výstup, je digitalizovaný v AD prevodníku $CONV_{IN}$ a je privedený na register veličiny signálu pamäťovej bunky CE_{DU} .

Doba trvania je nastavená programom procesora.

Informácia v pamäťovej bunke CE_{DE} je during t v,

kde t je doba trvania a v je veličina signálu. Doba trvania je teda známa. Čítač času T_{COUNT} je resetovaný centrálnou riadiacou jednotkou $\underline{5}$ pri spustení doby trvania. Súvisle zvyšovaný čas v čítači T_{COUNT} je porovnávaný s časom v časovom registri v pamäťovej bunke CE_{DU} čítačom času T_{COMP} . Doba trvania končí, keď tieto časy sú zhodné.

Pamäťová bunka CE_{LAST} bola počas tejto doby trvania vyprázdnená a pamäťová bunka CE_{NEXT} naplnená. Pri prechode na ďalšiu periódu obsah pamäťovej bunky CE_{DU} je premiestnený do pamäťovej bunky CE_{LAST} a veličina signálu v pamäťovej bunke CE_{NEXT} je presunutá do pamäťovej bunky CE_{DU} .

Doba trvania je nastavená prostredím.

Informácia v pamäťovej bunke je CE_{DU} je during $\$$ v, kde v je veličina signálu a $\$$ predstavuje ľubovoľnú dobu z možných dôb trvania, teda že tento výraz sa zjednotí s každou dobou trvania. Doba trvania je teda neznáma a prostredie ju určí. Čítač času T_{COUNT} je resetovaný externe na začiatku doby trvania. Súvisle zvyšovaný čas v časovom čítači T_{COUNT} je privedený na register času v pamäťovej bunke CE_{DU} . Keď prostredie vyšle ďalší externý resetovaný signál, doba trvania skončila. Takto externý resetovací signál riadi selektor SEL_1 , aby rozpojil spojenie k čítaču času buď priamo alebo pomocou nulovacieho signálu z čítača času.

Počas doby trvania bola pamäťová bunka CE_{LAST} vyprázdnená a pamäťová bunka CE_{NEXT} naplnená. Pri prechode na ďalšiu periódu je obsah v pamäťovej bunke CE_{DU} premiestnený do pamäťovej bunky CE_{LAST} a obsah v pamäťovej bunke CE_{NEXT} je premiestnený do pamäťovej bunky CE_{DU} .

Jednoduchý príklad.

Pre ilustráciu činnosti portu bude uvedený malý príklad. Povedzme, že chceme zistiť jednu z dvoch rôznych postupností, prvá postupnosť má každú druhú veličinu signálu určenú, napríklad hodnotou 17 a má určené všetky doby trvania, napríklad 1 sekundu a druhá postupnosť má všetky veličiny signálu určené, napríklad 1, 3, 3, 4 atď, ale nemá určenú žiadnu dobu trvania. Port, ktorý

má byť použitý, sa nazýva port_1 . Reprezentácia spínača:

unify (port_1 "interval behaviour"),

kde "interval behaviour" je operácia alt, ktorá určuje všetky alternatívne postupnosti, teda veľké datové štruktúry nasledujúceho druhu:

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

Teraz, pokiaľ postupnosť vstupného signálu môže byť unifikovaná vyššie uvedenou štruktúrou, vstup je prijatý a vykoná sa príslušná činnosť. Jeden príklad prijatej postupnosti je ((1s 17) (1s 0)(1s 17)(1s 99)(1s 17)), čo je postupnosť, ktorá môže byť unifikovaná prvou postupnosťou vnútorného chovania (interval behaviour), uvedenou vyššie. Iná postupnosť, ktorá bude prijatá, je ((2s 1)(4s 2)(1s 3)), ktorá bude unifikovaná druhou postupnosťou vnútorného chovania. Príkladom neprijateľnej postupnosti je ((1s 2)...), pretože veličina signálu, teda 2, nemôže byť unifikovaná veličinami signálu, danými v týchto dvoch postupnostiach vnútorného chovania.

Komunikácia s vonkajším svetom.

Dômyselnosť konštrukcie portu, že hodnoty môžu byť prenesené dovnútra alebo von z procesora v závislosti na tom, či signál je alebo nie je definovaný alebo či vnútorná hodnota je definovaná alebo nie (má hodnotu \$) je v tom, že procesor nie je obmedzený na systém jednoduchého vstupného a výstupného portu, ktorý je už popísaný. Preto je možné napísať program, v ktorom je port alternatívne zapísaný do a čítaný z procesoru. Príkladom toho je nasledujúci program:

unify(port "interval behaviour")

ktoré má vnútorné chovanie:

```
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 2)(during 1s 4)
        )
      )
    )
```

atď, to znamená

```
(seq(during input)(during output)(during input)(during output)
```

V tomto vnútornom chovaní prostredie odovzdá hodnotu a procesor odpovie tým, že odovzdá na výstup jej štvorec počas ďalšieho intervalu. Táto metóda dáva plnú kontrolu toho, ako dáta budú jednoduchým spôsobom postupovať.

Keď vstupná hodnota nie je unifikovaná, teda že nemôže byť upravená rovnako ako odpovedajúca hodnota v procesore, výsledok je nothing (nič). Takto sa bude postupovať rovnako až k operácii alt a celá vetva je vypustená z výrazu alt.

Reprezentácia vnútorného chovania má rovnakú dĺžku ako vstupná postupnosť. Časť kratšieho a výraznejšieho spôsobu ako vyjadriť vnútorné chovanie pomocou plnej sily funkčného jazyka je daná takto:

seq(during(1s X)

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

kde during(1s X) spracováva vstup veličiny signálu X a kde ďalšia štruktúra during, obsahujúca výraz lambda, pracuje s výstupom štvorca vstupnej hodnoty. Preto vyššie uvedená štruktúra môže byť súčasťou obcejnejšej štruktúry, ktorá je schopná s použitím rekurzcie ovládať viac ako jeden vstupný/výstupný pár. Výraz lambda je tu použitý na definíciu štvorcovej funkcie.

Port H.

Uvedený popis vstupného/výstupného portu môže byť generalizovaný tak, aby bol schopný pracovať so vstupnými a výstupnými dátovými štruktúrami namiesto štruktúr during. V tom prípade miesto unifikácie chovanie s externým chovaním pomocou unifikácie doby trvania a veličiny signálu, je možné vykonať unifikáciu bitových vzorcov. Prichádzajúci signál môže byť digitalizovaný. Ďalej prichádzajúci signál môže reprezentovať dátové štruktúry alebo programové štruktúry ako programy v jazyku H. Napríklad pole pamäťovej bunky v pamäťovej bunke v porte môže byť unifikované bitovými vzorcami na vstupe portu.

Príklad: Vnútorná štruktúra apply(X1 X2) môže byť unifikovaná bitovými vzorcami na vstupe tak, že konečná unifikovaná interná štruktúra bude: apply(+ list(1 2)), čo je štruktúra, uvedená na obr.8A a 8B. Tu X1 a X2 reprezentujú "any

bitpattern" (akýkoľvek bitový vzorec). X1 je unifikovaný bitovým vzorcom, ktorý reprezentuje inštrukčný kód pre "+" a X2 je unifikovaný bitovým vzorcom, ktorý reprezentuje "list(1 2)".

Cez H port, ktorý prijíma kód v jazyku H, môže byť program načítaný do objektového ukladania. Port H môže byť tiež použitý medzi procesormi, čo im umožní, aby prenášali programy alebo dáta alebo obidve. Prenesený kód programu môže byť na začiatku označený ako dáta, aby sa zabránilo jeho okamžitému vykonávaniu.

Redukčný procesor prvého rádu.

V krátkosti zhrnuté - redukčný procesor je riadený programom, ktorý má konkrétnu nie sekvenčnú štruktúru a redukuje túto štruktúru v niekoľkých redukčných krokoch vrátane rôznych druhov redukcii stromovej štruktúry. Objektové ukladanie 1 v procesore prvého rádu obsahujú množinu pamäťových buniek 10, z ktorých je každá schopná uložiť informáciu, ktorá by mohla vyvolať redukcii. Komunikačná sieť, usporiadanie pamätevej zbernice spoločne s hlavou spínača, riadený centrálnou riadiacou jednotkou 6, odovzdáva výsledok každej redukcie všetkým pamäťovým bunkám.

Komunikačná sieť je usporiadanie zbernice, ktorá má svoje riadiace linky a dátové linky, ktoré sú všetky pripojené na uvedené pamäťové bunky. Centrálna riadiaca jednotka 6 je spoločná pre všetky pamäťové bunky. Je to informácia v pamäťových bunkách 10, ktorá odovzdáva informáciu do centrálnej riadiacej jednotky 6, aká redukcia sa má vykonať a tým i aké riadiace slovo má vydať. Pretože centrálna riadiaca jednotka 6 je boolovské hradlové pole, informácia v pamäťových bunkách je v tvare vysokých a nízkych napätí, ktoré sú odovzdávané na hradlá AND a OR v zložitých konfiguráciách, ktoré vytvárajú zložité výstupy, zložené z vysokých a nízkych napätí, ktoré majú byť odovzdané na veľa riadiacich bodov v pamäťových bunkách.

Každá pamäťová bunka v objektovom ukladaní a v jadrovej bunke je takto schopná si zapamätať informáciu, potrebnú pre vykonanie redukčnej operácie. Redukčná informácia môže tiež

obsahovať referenciu na niektoré iné pamäťové bunky. Keď je vykonanie ukončené, teda štruktúra je, pokiaľ je to možné, čo najviac redukovaná, výsledok je v kanonickom tvare. Pokiaľ je kanonický tvar väčší ako môže byť uložený v jednej pamäťovej bunke, sú tu potom referencie na ostatné pamäťové bunky, kde je uložený zvyšok výrazu. Príkladom pre popísané zapojenie je zoznam s piatimi prvkami, ktoré vyžadujú dve pamäťové bunky, aby mohli byť uložené v objektovom ukladaní 1.

Jadrová bunka je schopná vykonať všetky druhy redukcií. Je nutné poznamenať, že počet jadrových buniek nie je obmedzený na jednu a tak môže byť niekoľko jadrových buniek i keď nie sú v zapojení uvedené. Pamäťové bunky v objektovom ukladaní sú schopné vykonávať iba obmedzené časti niektorých zo všetkých druhov operácií.

Niekoľko jadrových buniek tu môže byť na to, aby pracovali paralelne s kompletnými štruktúrami v tvare kompletného stromu spínača alebo častiach štruktúry v tvare častí stromu spínača. Pretože výsledok niekoľkých redukcií sa nevzťahuje na poradie redukcie, vo vnútri prekrývajúcich sa kontextov môžu pracovať rôzne jadrové bunky. Všeobecne nie je nutné vykonávať žiadne opatrenie na to, aby sa zaistila integrita dát - je zaistená automaticky tým, ako procesor podľa vynálezu pracuje. Napríklad keď dve jadrové bunky redukujú dve rovnaké časti stromu spínača, jediný možný následok je ten, že sa vykoná nepotrebná redukcia - výsledok je rovnaký.

Komunikačná sieť je prispôsobená tiež na prenos informácie z bunky objektového ukladania, ktorá je vybratá z buniek objektového ukladania do jednej z jadrových buniek, ktorá je vybratá komunikačnou sieťou a z vybratej jadrovej bunky do jednej alebo niekoľkých buniek objektového ukladania, ktoré sú vybraté komunikačnou sieťou.

Komunikačná sieť vykonáva operáciu prenosu medzi jadrovou bunkou a ktoroukoľvek pamäťovou bunkou tak, že nahradí obsah v prijímacej časti obsahom z vysielacej časti. Dvojcestný prenos je realizovaný prehodením obsahu dotknutých buniek. Každý bunkový

spínač, uložený v pamäťových bunkách, obsahuje znak vykonateľnosti, polohy v stromovej štruktúre, identifikátora, prostredia a strom hodnôt alebo určenie pre ďalšie pamäťové bunky.

Strom hodnôt, ktorý je obsiahnutý vo viacúrovňovom spínači obsahuje lístkové prvky, to znamená konečné prvky a zložené prvky, kde každý zložený prvok bude zahrňovať typ a zoznam hodnôt. Označenie vykonateľnosti v spínači má minimálne dva stavy, z ktorých prvý je kľudový stav a druhý stav je vykonávateľný stav. Označenie polohy v spínači má minimálne stavy, z ktorých prvý stav je poloha uzla a druhý stav je pozícia koreňa.

Minimálne niektoré označenia spínača sú prezentované v rôznych stavoch, z ktorých každý je zložený z niekoľkých bitov, reprezentujúcich binárny kód.

Každý prvok hodnota/označenie v bunkovom spínači je zložený z príznakového slova (tag) a slova num, čo je bitový vzorec, zložený z niekoľkých bitov v jednotke alebo nule. Príznaky sú rozdelené do nepriamych tried a priamych tried. Slovo num môže napríklad byť typu integer alebo s plávajúcou rádovou čiarkou. Typ integer je uložený v prvku v binárne kódovanom formáte (tiež nazývaný reprezentácia) tak, že všetky hodnoty sú usporiadané od najnižších po najvyššie hodnoty v sérii čísiel, kde každé číslo obsahuje niekoľko bitových hodnôt a v ktorom je nula uprostred radu čísiel ako binárne číslo, ktoré má binárnu hodnotu true na pozícii najvýznamnejšieho bitu a ktoré má binárnu hodnotu vo zvyšku svojich bitov. Číslo v plávajúcej rádovej čiarky je uložené v prvku v binárne kódovanom formáte (reprezentácii) tak, že obsahuje znamienko, znamienko exponentu a kódové pole, pole exponentu a pole mantisy, pričom znamienko, znamienko exponentu a kódové pole majú indikáciu rozdelenia pozície medzi uvedené pole exponentu a uvedené pole mantisy tak, že pole exponentu a mantisy majú premenlivú dĺžku.

Procesor podľa vynálezu teda používa usporiadanie kódov číselných hodnôt do reprezentácie hustoty, to znamená každá

kódovaná reprezentácia hodnoty odpovedá iba interpretovanej hodnote. Prvý druh kódovania sa používa v binárnom slove, ktoré reprezentuje typ integer. Ďalší druh kódovania sa používa pre binárne slovo, ktoré predstavuje plávajúcu rádovú čiarku. Kódovania sú také, že po kódovaní s plávajúcou rádovou čiarkou sú v rovnakom poradí ako pre typ integer. Reprezentácia hustoty je vytvorená zahrnutím virtuálnej "1", teda "1", ktorá nie je daná fyzicky v reprezentácii pred polom slova pre časť mantisy hodnoty s plávajúcou rádovou čiarkou. Reprezentácie číselnej hodnoty sú rozdelené na bitové skupiny, nazývané skupiny presnosti, kde každá má dopredu definovaný počet bitov, rozdelenie medzi časť mantisy a exponentu, kde reprezentácia predstavuje hodnotu s plávajúcou rádovou čiarkou, ktoré je vykonané medzi dve skupiny presnosti.

Procesor druhého rádu (SOP).

Ako je uvedené na obr. 36, na FOPN môže byť navzájom pripojených niekoľko procesorov prvého rádu FOP1 cez sieť NET, pričom prepojené procesory prvého rádu predstavujú procesor druhého rádu.

Pripojenie na sieť NET môže byť vykonané ako prostriedok pre prenos dát ako prostriedok 4 na obr.1, ako každý procesor prvého rádu FOPi, kde i je celé číslo medzi 1 a N. Prostriedok 4 pre prenos dát na obr. 1 obsahuje rad registrov R_{id} , R_{env} , R_{v0} , R_v , R_{v1} , R_{v3} . Každý register v sieti je pripojený na vertikálnu zbernicovú linku, ktorá je označená indexom. Registre siete sa používajú, keď je niekoľko objektov ukladania, vrátane jadrových buniek, ktoré sú pripojené tak, aby sa vykonávali paralelne. Komunikácia do a zo všetkých týchto objektových ukladaní sa vykonáva pred registrami siete.

Teoreticky zdieľajú všetky procesory prvého rádu rovnakú informáciu M, ktorá je celkovou oblasťou systému. Je nutné, aby zjednotenie všetkých redukčných procesorov prvého rádu sa rovnalo oblasti systému M. Avšak každý procesor prvého rádu spracováva informáciu, ktorá je uložená v jeho vlastnom objekte ukladania.

To znamená, že jednoduchý spínač je možno nájsť vo viac ako jednom procesore. Každý procesor je schopný sledovať, v ktorom procesore prvého rádu je spínač uložený. Jednoduchý spínač v niekoľkých objektových uloženiach je zhodný - to je zabezpečené redukčným mechanizmom.

Sieť spínača obsahuje spínač a jeho otca. Je možné posunúť sieť spínača posunutím spínačov medzi procesory. Aby sa minimalizovala komunikácia, žiada sa udržať sémanticky viazané spínače spoločne v jednom alebo v malom počte procesorov. Komunikačný protokol môže byť taký, aby zaistil presun spínačov počas doby, kedy sa nevyžaduje žiadna komunikácia medzi procesormi. Strom spínača môže byť redukovaný paralelne v rôznych procesoroch. Pre tieto operácie nie je nutná žiadna synchronizácia. Preto redukcia častí stromu spínača môže byť vykonaná v rôznych okamžikoch. Nepríde ku žiadnej chybe, pretože výsledok redukcii je rovnaký bez ohľadu na to, kedy je vykonaný.

Samozrejme, redukcia spínača, ktorý má v ostatných procesoroch rovnaké kópie, sa môže vykonať iba v jednom FOPi (i je celé číslo medzi 1 a N) paralelných procesorov FOP1 až FOPN a výsledok redukcie je distribuovaný na všetky procesory, ktoré sú spojené v sieti NET pokiaľ možno čo najskôr, aby sa nestrácali prostriedky procesoru. Bude však obyčajne vhodnejšie a tiež ekonomickejšie na čas, keď sa budú vykonávať paralelné redukcie v rôznych častiach procesorov prvého rádu. Napríklad sieť NET môže byť prechodne obsadená obsluhovaním niektorého z procesorov prvého rádu, ktorý je na ňu pripojený a medzitým sa môžu vykonať paralelné redukcie.

Na skutočný druh siete NET, ktorá má byť zaistená, by sa nemalo pozerať ako na súčasť vynálezu a bude preto popísaná podrobne neskôršie. Niektoré príklady rôznych druhov siete budú uvedené ďalej. Je mnoho typov komunikačných sietí, ktoré majú rôzne druhy smeru ciest a stratégií prepojovania, ktoré je možné použiť. Sieť NET môže byť v tvare kruhovej siete, ktorá má jednoduché alebo dvojité spoje, sieť do hviezdy, plne zapojená sieť alebo podobne.

Aby sa zabránilo príliš nadmernému prepojovaniu medzi procesormi prvého rádu FOP1 k FOPN, prenos dát medzi nimi môže

byť vykonávaný sériovo na sériovej zbernici. Prenos dát je vykonávaný s distribuovaným riadením. Teda nie s použitím centrálného riadiaceho zariadenia, ktoré riadi všetky procesory prvého rádu, ďalej nazývané FOP. Jeden z FOP vyžaduje prístup na sieť NET na konkrétnej zbernici DEM a potom sú ostatné FOP nastavené do stavu príjmu. Ale prenos signálu ako žiadosti o prístup môže byť zaistený iným spôsobom ako cez samotnú zbernicu, napríklad ako kľudový signál, ktorý je prenesený na zbernicu, ktorá je popísaná v súbežnej prihláške. Dáta medzi procesorom prvého rádu môžu byť prenesené a uložené v registroch v prostriedkoch pre prenos dát v FOP, zatiaľ čo sa v FOP vykoná redukčná operácia. Registrové bunky v prostriedkoch pre prenos dát sú z toho dôvodu riadené sekvenčne centrálnou riadiacou jednotkou z FOP.

Synchronizácia medzi procesormi FOP sa vykonáva synchronizáciou hodinových signálov v každom FOP vo vzájomnom vzťahu. Taká synchronizácia je spoločná a nebude preto ďalej popisovaná.

Ako alternatíva však môže byť v centrálnaj riadiacej jednotke vytvorený procesor druhého rádu, ktorý riadi komunikáciu medzi procesormi FOP. Táto riadiaca jednotka môže byť boolovské hradlové pole s centrálnymi riadiacimi jednotkami vo všetkých procesoroch prvého rádu. Nedoporučuje sa ale mať takú druhú riadiacu jednotku.

Obr. 36 ukazuje veľmi všeobecný redukčný procesor druhého rádu podľa vynálezu, v ktorom môže prebiehať prenos dát ľubovoľného druhu medzi procesormi FOP.

Prvé zapojenie redukčného procesoru druhého rádu (SOP).

Podľa konkrétneho zapojenia procesoru druhého rádu, ktorý je uvedený na obr. 37A a ktorý je prispôsobený pre riadenie prenosu distribuovaných dát medzi procesormi FOP, môže byť niekoľko procesorov FOP $FOP_{1,1}$ až $FOP_{M,M}$ zapojených do štvorcového poľa. Každý FOP v uvedenom štvorcovom poli obsahuje kanál CAN pre prenos dát, ktoré idú do susedného redukčného procesoru prvého rádu v štvorcovom poli. Ako je uvedené na

obr.37B, oddelené prostriedky pre prenos dát znamenajú, že DTF1 až DTF4 môžu byť zaistené pre každý kanál prenosu CAN a tiež pre každý port, pripojený na FOP. Tým sa umožní súčasný prenos dát z FOP na niekoľko ďalších procesorov FOP.

V type štruktúra, ktorá je uvedená na obr. 37A, každý procesor FOP komunikuje iba najviac so štyrmi najbližšími procesormi FOP a najradšej cez samostatný kanál ku každému. Medzi každým párom prostriedkov pre prenos dát je potrebná iba jednoduchá cesta pre riadenie pri vzájomnej komunikácii medzi nimi, kde na tomto prostriedku je nastavený príznak, keď komunikácia vyžaduje nastavenie ostatných procesorov FOP, ktoré sú pripravené na prenos dát vo svojich prostriedkoch pre prenos DFTn.

Siete spínačov a oblasti FOP.

Siete sú priradené rôznym procesorom FOP. Ako už bolo popísané vyššie, bunkový spínač je označený strojovým identifikátorom, napríklad #5. Bunkový spínač môže obsahovať nepriame prvky, identifikátory, ktoré ho spájajú s ostatnými bunkovými spínačmi. Všetky bunkové spínače, ktoré obsahujú identifikátory, vzťahujúce sa k určitému spínaču, tvoria sieť spínača, teda že obsahuje spínač a jeho otca. Oblasť je typický štvorec procesorov FOP, takže oblasť REG je obklopená súvislou priamkou v obr. 37A. Táto oblasť takto obsahuje všetky spínače, ktoré majú identifikátor #5. Avšak bunkový spínač s identifikátorom #5, ku ktorému sa viažu spínače s prvkami identifikátora #5, môžu na druhej strane obsahovať prvky identifikátora, ako #2 a #9, ktoré sa nachádzajú v procesoroch FOP mimo oblasť identifikátora #5. V popise obr. 37C nižšie bude uvedené viac o oblastiach.

Jeden dôvod pre zavedenie oblastí je, že asociatívne vyhľadávanie a nahradzovanie môže byť obmedzené v tom zmysle, že niekoľko zo všetkých objektových ukladaní procesorov FOP potrebuje byť prehľadaných, aby sa vykonala aktualizácia otcov redukovaných spínačov - iba objektové ukladania, ktoré patria procesorom FOP v oblasti, potrebujú byť prehľadané.

Komunikácia s okolitým svetom.

Pozdĺž minimálne jednej hrany štvorcového poľa procesorov FOP je vytvorených niekoľko portov PORT pre vstup/výstup dát z a do procesoru druhého rádu. Každý port je jednotka, ktorá zaistuje komunikáciu s externým prostredím procesoru. Používaný port môže byť takého druhu, ktorý je popísaný vyššie a ktorý vykonáva vstupné/výstupné operácie cez unifikáciu.

Cez iný port, port H, ktorý prijíma kód v jazyku H, môže byť načítaný program do objektových ukladaní v procesoroch FOP. Port H môže byť tiež použitý medzi procesormi, čo im umožní presunúť programy alebo dáta alebo obidve.

Iný druh portu môže existovať pre vstup textu v ASCII, ktorý umožní, aby procesor podľa vynálezu komunikoval so zariadením, ktoré je schopné vyslať a/alebo prijať ASCII text, napríklad konvenčný počítač. Tento druh portu nie je súčasťou daného vynálezu a nebude preto ďalej podrobnejšie popisovaný.

Mäkké a tvrdé okruhy.

Tvrdý okruh je "na tvrdo zadrôtovaná" komunikačná cesta, teda cesta, ktorá je daná hardwarom. Mäkký okruh je "na mätko zadrôtovaná" komunikačná cesta, teda cesta, ktorá je daná na abstraktnej úrovni, ktorá sa s časom môže meniť. Prirodzene mäkký okruh využíva hardware pre aktuálnu komunikáciu. Cesta, ktorá obsahuje každý procesor, ktorý obsahuje špecifickú spínačovú sieť, teda s identifikátorom #5, sa nazýva mäkký okruh.

Oblasť pre sieť spínača je podsúborom všetkých procesorov. Sieť spínača, teda spínač a jeho otcovia, je možné nájsť v jej oblasti. Cieľom je vytvoriť oblasť pre každý spínač pokiaľ možno čo najmenšiu, aby sa obmedzilo množstvo komunikácií medzi procesormi.

Presný mäkký okruh.

Oblasť môže byť zložená tak, ako je uvedené na obr. 37D, z niekoľkých procesorov FOP na obr. 37A, ktoré majú identifikátory spínačov ID_{def} uložené na rôznych miestach v ich objektovom ukladaní, označené na obr. 37D malou horizontálnou

čiarou.

Pokiaľ tento spínač má byť nahradený, potom všetky procesory FOP, ktoré majú tento identifikátor spínača (otcovia), musia byť aktualizované cez mäkký okruh www_1 , ktorý obsahuje procesory FOP s identifikátorom ID_{def} . Veľkosť mäkkého okruhu www_1 je závislá na implementácii. V zapojení na obr. 37D má mäkký okruh, ktorý spojuje procesory FOP, obsahujúce aktuálnu sieť spínača, tvar serpentíny. Presné mäkké okruhy sú rôzne pre rôzne siete spínačov.

Oblastné mäkké okruhy.

Alternatívou k presným mäkkým okruhom, ktoré boli popísané vyššie a kde sieť spínača a jej mäkký okruh pokrývali rovnaké procesory FOP, je mať mäkké okruhy, ktoré budú obsahovať procesory FOP, ktoré nemusia nezbytné presne korešpondovať so sieťami spínačov, ktoré obsahujú. Namiesto toho mäkký okruh môže obsahovať každý FOP v oblasti. Obr. 36E ukazuje oblasť a znázorňuje, že aktualizovaná správa je vyslaná na každý FOP v oblasti siete spínača, ktorá je tvorená identifikátorom ID_{def} , teda že mäkký okruh www_2 obsahuje každý procesor FOP v oblasti. Oblastné mäkké okruhy (www_2) sú preferované, pretože objem účtovníctva (réžia) je menší ako keď sa používajú presné mäkké okruhy (www_1). Oblastný mäkký okruh pre špecifickú oblasť je rovnaký pre každú sieť spínača vo vnútri oblasti.

Základný tvar oblastnej siete sú okruhy a zbernice. Okruh môže dopraviť hodnotu na jeden obrat v slučke. Počas takejto operácie vykonáva rovnakú sémantiku ako zbernice v jednom cykle.

V okruhu môže byť v jednom momente vyslaných niekoľko inštrukcií spínača. Keď správa prešla okruhom raz dokola, môže byť vypustená.

Druhé zapojenie redukčného procesoru druhého rádu.

Podľa druhého zapojenia, ktoré je uvedené na obr. 38, je niekoľko procesorov FOP zapojených v hierarchii sietí, kde každá sieť je zbernica NET_1 až NET_N . Hlavný rozdiel v zapojení podľa obr. 37A je ten, že na obr. 37A je FOP cez kanály pre prenos dát

zapojený k väčšine štyroch iných procesorov FOP, než ako to je na obr. 38, kde je niekoľko procesorov FOP navzájom prepojených cez komunikačnú zbernicu. Distribuované riadenie je však možné dosiahnuť pomocou jednoduchého dekodéru priority (nie je uvedené) ku každej zbernici, aby sa nastavila priorita medzi procesormi FOP v prípade, že viac ako jeden žiada o prenos dát na jeden alebo niekoľko ostatných. Tento dekodér priority môže mať prakticky rovnakú konfiguráciu ako je uvedené na obr. 21A a 21B, čo je dekodér priority 11 podľa obr. 2 a 3.

Keď sa v niektorom procesore FOP vykoná redukcia, otcovia redukovaného spínača musia byť aktualizovaní. Procesor FOP, ktorý vykonal redukciu, odovzdá na zbernicu aktualizáciu správu, teda na podsúbor zberníc NET_1 až NET_A , ktoré patria oblasti danej siete spínača. Všetky procesory FOP, ktoré sú pripojené na zbernicu s aktualizovanou správou kontrolujú, či obsahuje časť siete spínača, kedy je potom časť aktualizovaná alebo prepísaná. Keď sa redukcia odovzdá na zbernici, je to tiež poznačené v lokálnom procesore FOP ako unifikácia. Pretože niekoľko procesorov FOP vykonáva redukciu súčasne, aktualizácia správa môže byť multiplexovaná podľa komunikačného protokolu. Pretože výsledok sa nevzťahuje na redukčný rád, komunikačný rád zbernice nie je nutný. Spoločné komunikačné protokoly zbernice môžu byť použité na to, aby sa zistilo, že zbernica bude mať v danom momente jeden vysielací procesor FOP.

Porty $PORT_2$, $PORT_{31}$, $PORT_{32}$ môžu byť pripojené na vybrané NET_2 a NET_3 medzi zbernicami. Ako je uvedené, je možné spojiť niekoľko portov na rovnakú zbernicu $PORT_{31}$, $PORT_{32}$.

Tretie zapojenie redukčného procesoru druhého rádu.

Podľa tretieho zapojenia podľa obr. 39 je niekoľko procesorov FOP zapojených v hierarchii sietí, kde každá sieť je tvrdý okruh $RING_1$ až $RING_4$, teda okruh, kde sú zúčastnené procesory FOP pevne prepojené navzájom. Oblasť môže pokryť jeden z niekoľkých okruhov. Najlepšie sa mäkký okruh zhoduje s tvrdým okruhom.

Kombinácie rôznych druhov procesorov SOP.

Ako je uvedené na obr. 40, je tiež možné kombinovať zapojenie, uvedené na obr. 37A, 38 a 39 tak, aby sa zapojili redukčné procesory druhého rádu v hierarchii sietí, ktoré majú minimálne dva druhy siete, prvý druh je sieť BUSNET a druhý je okruh RINGNET a tretí druh je štvorcová sieť SQUARENET.

Počítač druhého rádu podľa vynálezu takto môže mať siete, ktoré sú ako okruhy a/alebo zbernice. V počítači, kde bol redukovaný podstrom spínača, musia byť otcovia redukovaného syna informovaní o tom, že syn bol redukovaný. Pokiaľ sa stane, že tu bude také dobré usporiadanie, že syn a jeho otcovia sú umiestnení v okruhu procesorov FOP alebo v procesoroch FOP na zbernici, potom informácia, týkajúca sa redukcie syna, musí byť vyslaná iba cez tento okruh alebo túto zbernicu. Protokol počítača je preto najlepšie taký, že tento spínač, ktorý reprezentuje otca, je presunutý medzi procesory FOP takým spôsobom, aby boli dobre prístupné od svojho syna alebo synov, teda zo všetkých častí siete spínača by mali byť spoločne v malej oblasti.

Strojový protokol.

Protokol má konfiguračnú časť, ktorá zavádza zoznam portov a im priradené úlohy. Porty do procesoru podľa vynálezu, či ide o procesor prvého, druhého alebo nejakého vyššieho rádu, budú priradené určité časti aplikácie procesoru.

Protokol, ktorému je daná prednosť, obsahuje jednu časť s konfiguračnou portov, jednu časť, ktorá priraduje ciele procesorom FOP, jednu časť, ktorá realizuje redukčnú aritmetiku, jednu časť, ktorá realizuje ľubovoľne dlhý zoznam a jednu časť, ktorá realizuje ľubovoľne dlhé bitové pole hodnôt. Časti protokolu môžu byť obsiahnuté v hardware. Je to protokol, ktorý spúšťa synchronizáciu hodín v procesoroch FOP. To je najlepšie urobené v určitých časových intervaloch, dostatočne krátkych na to, aby hodiny boli synchronizované.

Chovanie je dané ako spínač. Procedúra priradenia je riadená určitou časťou protokolu. Priradenie alebo nové priradenie spínača môže prebiehať často medzi ostatnou realizáciou redukcie. Každý

procesor FOP môže vydať príkaz na presunutie spínača, pričom žiada iný FOP, aby uložil spínač, ktorý má byť presunutý.

Keď má byť presunutý spínač, napríklad keď objektové ukladanie v procesore je plné, procesor FOP, ktorý obsahuje spínač, odovzdá správu na zbernicu alebo okruh a žiada pritom o odpoveď od ostatných procesorov FOP, pripojených ku zbernici alebo okruhu, pokiaľ spínač môže byť do ľubovoľného z týchto procesorov FOP. Pokiaľ všetky tieto FOP dajú zápornú odpoveď, čo znamená, že žiadny z nich nemôže prijať spínač, tento spínač je podržaný v pôvodnom FOP. Pokiaľ jeden z odpovedajúcich procesorov FOP dá kladnú odpoveď, spínač je presunutý do tohoto FOP. Pokiaľ dá kladnú odpoveď niekoľko procesorov FOP, rad priority vyberie jeden FOP ako prijímač presunutého spínača. Rad priority môže byť napríklad geometrický rad ako rad najbližšieho suseda.

Oblasti.

Obr. 37 ukazuje zapojenie zväčšeného štvorcového poľa procesoru druhého radu, ktorý má 256 procesorov FOP. Kanály medzi procesormi a možnými portami do procesoru druhého radu nie sú uvedené. Každý FOP predstavuje oblasť veľkosti nula. Obr. 37C ukazuje, že štvorcové pole procesoru druhého radu môže byť logicky (nie fyzicky) rozdelené na štvorcové oblasti rôznych veľkostí, umiestnených po strane každého pravidelného vzorca. Takto má každá oblasť rozmer $2^n \times 2^n$ procesorov FOP. Oblasti sú umiestnené disjunktne, teda že sa naprekrývajú. Protokol však môže byť taký, že oblasť môže byť tiež definovaná tak, že bude umiestnená polovicou oblasti v rade alebo stĺpci alebo v rade i v stĺpci.

Protokol je taký, že sleduje oblasť a využíva znalosti, v ktorej oblasti alebo oblastiach je umiestnený otec alebo synovia, keď sa vykonáva vyhľadávanie a nahradenie redukovaným spínačom. To je urobené tak, že bitový vzorec pre nepriame prvky, teda identifikátory, umiestnený v poliach hodnota/označenie bitového vzorca, bude obsahovať časť bitového vzorca, ktorý udáva veľkosť oblasti, ku ktorej patrí a časť bitového vzorca, ktorý udáva, či by mala byť oblasť niektorým smerom premiestnená. Bity smeru

premiestnenia by mohli byť napríklad dva, jeden keď bude "1", bude udávať posunutie doprava a druhý keď bude "1", bude udávať posunutie smerom hore. Keď sú obidva "1", potom posunutie bude polovica kroku doprava a polovica kroku smerom hore.

Keď sa vykoná redukcia syna, informácia sa pošle cez jeho oblasť(i), kde sú umiestnení otcovia, v mäkkom okruhu, ktorý obsahuje všetky procesory FOP v oblasti. To znamená, že pokiaľ je oblasť veľmi veľká, mäkký okruh bude veľký a distribúcia informácie o redukovanom synovi bude vyžadovať čas a prostriedky počítača. Preto je protokol taký, aby sa oblasti siete spínača zmenšili a aby boli pokiaľ možno čo najmenšie. Preto pokiaľ je vzdialenosť medzi synom a otcom veľká, strojový protokol je taký, aby sa nastavila vyššia priorita na presunutie týchto spínačov namiesto spínačov, kde vzdialenosť otca a syna je malá. Je nutné poznamenať, že operáciu zmenšenia oblasti nie je možné vykonať neobmedzene. Niektoré objektové ukladania môžu byť napríklad úplne zaplnené dátami. Niektorí z otcov môžu mať ostatných synov umiestnených ďaleko od synov, pre ktorých sa operácia zmenšenia vykonáva.

Protokol štvorcového poľa procesora druhého rádu bude s výhodou taký, aby sa koncentrovali sémanticky súvisiace dáta do malej oblasti. Je tiež potrebné presunúť dáta, aby sa priradili určitej oblasti pre konkrétne druhy činnosti a aby sa presunuli dáta, ktoré majú vzťah k týmto činnostiam smerom k oblastiam, ktoré sú im priradené. Príkladom činnosti môže byť konkrétny druh výpočtu. Iný príklad činnosti je činnosť, ktorá má vzťah ku vstupnému/výstupnému portu, ktorý môže mať svoje vnútorné chovanie pokryté oblasťou.

Redukčný procesor tretieho rádu.

Spínače sú takto presúvané dokola v redukčnom procesore druhého rádu podľa vynálezu. Redukčný procesor však nemusí byť iba v jednej geografickej polohe. Rovnaký procesor môže mať napríklad jednu časť vo Švédsku a druhú v Austrálii. Preto môže byť vytvorený procesor tretieho rádu, ktorý je na obr. 41 a ktorý obsahuje niekoľko procesorov druhého rádu. Na obr. 41 sú ukázané

tri procesory SOP1, SOP2 a SOP3. SOP1 zahrňuje kombináciu procesora SOP so štvorcovým polom a procesorom SOP so zbernicou.

Asociatívne adresovanie a fyzické adresovanie.

Je potrebné minimalizovať objem komunikácie medzi rôznymi geografickými miestami, pretože náklady a rýchlosť prenosu dát bude stúpať so vzdialenosťou medzi miestami. Asociatívne vyhľadávanie by vyžadovalo komunikáciu na dlhú vzdialenosť, pretože každý uzol v systéme sa musí zúčastniť vyhľadávania. Je výhodné v tomto prípade prijať fyzické adresovanie, aby sa udržala komunikácia na dlhú vzdialenosť na minimu. Fyzické adresovanie v tomto prípade znamená, že adresujeme fyzické miesto, teda že poznáme prijímač dopredu. Záverom by bolo výhodné prijať asociatívne adresovanie vo vnútri procesora SOP alebo skupiny geograficky združených procesorov SOP a prijať fyzické adresovanie medzi takými procesormi alebo ich skupinami. Ako je zrejmé z uvedeného, asociatívne adresovanie vo vnútri procesora SOP môže byť obmedzené použitím oblasti.

Stále ale máme situáciu, kde redukcia syna bude hlásená všetkým jeho otcom. V procesore druhého rádu je toto urobené s použitím oblastí, pričom každá sieť je umiestnená vo vnútri pokiaľ možno čo najmenšej oblasti. V procesore tretieho rádu je jedno kritické hľadisko navyše, s ktorým je nutné uvažovať - globálny adresovací aspekt.

Globálne adresovanie.

Miesta objektového ukladania sú "adresované" cez obsah miesto fyzickej adresy, to znamená, že sa používa asociatívne adresovanie. Tento prístup nemá dávať prednosť globálnej úrovni z dôvodu objemu komunikácie na dlhú vzdialenosť, ktorá by bola vyžadovaná. Globálne adresovanie je tak ovládané viac konvenčným spôsobom. Miesto toho, aby sa vykonávalo globálne asociatívne vyhľadávanie a nahradzovanie keď je redukovaný syn, vykonáva sa lokálne, to znamená vo vnútri procesoru druhého rádu v ktorom je syn umiestnený, asociatívne vyhľadávanie a nahradzovanie. Ďalej je vykonávané globálne adresovanie vtedy, keď to je nutné,

napríklad keď otec redukovaného syna je umiestnený v inom procesore SOP na inom geografickom mieste. Globálne adresovanie nie je asociatívne, namiesto toho sa používa fyzické adresovanie. Fyzická adresa určuje SOP, ktorý by mohol obsahovať mnoho rôznych procesorov SOP. Prenesená informácia obsahuje globálnu adresu na synovský spínač id_0 , to znamená, že identifikátor id_0 je jednoznačný medzi procesormi v systéme. Informácia tiež obsahuje redukovaného syna. V každom geograficky samostatnom procesore SOP je dátová štruktúra, ktorá obsahuje informáciu o synoch, ktorí majú otca mimo SOP a každý syn má zoznam nelokálnych otcov. Táto dátová štruktúra bude ďalej nazývaná globálny spínač C_0 . Globálny spínač C_0 môže obsahovať ďalšiu informáciu.

Je niekoľko spôsobov na vykonanie globálnej aktualizácie. Globálna aktualizácia znamená, že minimálne jeden otec je umiestnený v inom procesore SOP ako syn.

Globálna aktualizácia bez prekladu adresy.

Keď je spínač redukovaný, vykoná sa vyhľadávanie identifikátora v globálnom spínači C_0 , ktorý označuje redukovaný spínač. Keď je identifikátor nájdený, redukovaný spínač (syn) má aspoň jedného nelogického mimo SOP - je nutná globálna aktualizácia. Globálna aktualizácia nie je nutná, pokiaľ identifikátor nie je nájdený. V oboch prípadoch sa vo vnútri procesoru SOP vykoná asociatívne vyhľadávanie a nahradenie. Pokiaľ globálna aktualizácia je nutná, prehľadá sa zoznam nelogických otcov v globálnom spínači C_0 . Pre každého otca v zozname je von vyslaná správa do SOP, ktorý obsahuje otca. Správa obsahuje globálnu adresu na redukovaný spínač a informáciu o redukcii samotnej. Prijímajúci procesor SOP vykoná asociatívne vyhľadávanie a nahradenie, vymení každý výskyt globálneho identifikátora s výsledkom redukcie. Tento protokol vyžaduje, aby každý identifikátor bol jednotný, to znamená, že bitový vzorec pre identifikátor je jednotný vo všetkých procesoroch v systéme.

V zapojení tohto protokolu je každému procesoru SOP priradené jednotné číslo a každý identifikátor má číslo SOP zakódované v bitovom vzorci, čo znamená, že fyzická adresa, ktorá

identifikuje SOP, je implicitná v bitovom vzorci identifikátora. Každý SOP "vlastní" rozsah identifikátorov a keď má byť do systému začlenený nový SOP, je mu priradený nový rozsah, vo vnútri ktorého nie sú žiadne identifikátory.

Príklad: SOP1 má rozsah identifikátora 100 - 199, SOP2 má rozsah identifikátora 200 - 299, ... , SOP8 má rozsah identifikátora 800 - 899. Do systému má byť začlenený nový SOP - bude mať priradené číslo 9 a rozsah 900 - 999. Na obr. 42A sú uvedené časti príkladu štruktúry. Na obr. 42A sú uvedené časti procesoru tretieho rádu s procesormi SOP1, SOP6 a SOP9. Na obr. 42C je uvedená časť globálneho spínača. Syn, ktorý je určený identifikátorom 623, má byť redukovaný. Lokálne asociatívne vyhľadávanie a nahradenie má byť vykonané zamenením identifikátora 623 s výsledkom, ktorý označuje. V globálnom spínači C_9 v SOP6 môže byť nájdený syn s nelokálnymi otcami, ktoré sú označené 121 a 999 ako prvky zoznamu (viď obr. 42C). Správa je vyslaná do SOP1 a SOP9 s informáciou o redukcii, teda redukovaný identifikátor, výsledok redukcie, atď. SOP1 a SOP9 vykonávajú vyhľadávanie a nahradenie výmenou identifikátora 623 s výsledkom, ktorý označuje. Dátová štruktúra v globálnom spínači C_9 môže byť teraz modifikovaná vynechaním zoznamu globálnych otcov pre daného syna.

Namiesto toho, aby sa uložil zoznam nelokálneho otca, môže byť použitý zoznam SOP čísiel, ktorý obsahuje otca.

Globálna aktualizácia s preložením adresy.

V tomto prípade rozlišujeme medzi globálnou a lokálnu adresou, pričom lokálna adresa je jednoznačná vo vnútri SOP a globálna adresa je jednoznačná medzi všetkými procesormi SOP v systéme. Hlavný rozdiel medzi týmto prístupom a prístupom uvedeným vyššie je to, že pre tento prístup je nutný preklad adresy.

S prekladom adresy môže pracovať globálna štruktúra C_9 . Spínač C_9 môže obsahovať informáciu o rozsahu identifikátorov, ktoré SOP vlastní. Ďalej môže použiť túto informáciu na preklad globálnej adresy na lokálnu adresu.

Aktuálny prenos dát a komunikačný protokol pre procesor tretieho rádu môže byť navrhnutý tak, že bude využívať princípy,

ktoré už boli zavedené v poli dátovej komunikácie.

Prirodzene iný redukčný procesor tretieho rádu by mohol byť urobený za použitia konvenčnej technológie okrem vyššie uvedeného popisu za použitia procesora prvého rádu, druhého rádu a/alebo tretieho rádu ako stavebných blokov.

Zatiaľ čo vynález bol popísaný s odkazom na špecifické zapojenie, skúsení čitatelia pochopia, že je možné vykonať rôzne zmeny a navrhnúť iné ekvivalenty bez toho, aby prišlo k veľkej zmene od skutočnej myšlienky a cieľa vynálezu. Okrem toho je možné vykonať zmeny bez odklonu od základnej technológie vynálezu.

P A T E N T O V É N Á R O K Y

1. Redukčný procesor riadený programom, ktorý má štruktúru a bol upravený tak, aby prijal túto štruktúru v niekoľkých redukčných krokoch vrátane rôznych druhov redukcíí, v y z n a - č u j ú c i s a t ý m, že procesor prvého rádu tohoto druhu obsahuje aktívne ukladanie (1, 2), okrem toho tiež obsahuje a.množinu aktívnych pamäťových buniek (10, 2), z ktorých každá má možnosť vyvolať vykonanie redukčnej operácie a
b.komunikačnú sieť ($t_1, t_2, id, env, v_0, v_1, v_2, v_3, 12, 13, 14, 6, 7, 11, 16, 17$), ktorá odovzdáva výsledok každej redukcie do všetkých buniek medzi všetkými pamäťovými bunkami, ktoré majú spojenie na uvedený výsledok.
2. Redukčný procesor podľa nároku 1, v y z n a č u j ú c i s a t ý m, že uvedená komunikačná sieť obsahuje usporiadanie zbernice, ktorá má riadiace linky (linky od a do 6) a datové linky ($t_1, t_2, id, env, v_0, v_1, v_2, v_3$), kde sú všetky linky pripojené ku všetkým uvedeným pamäťovým bunkám a prostriedky riadenia (6), ktoré sú spoločné pre všetky uvedené pamäťové bunky.
3. Redukčný procesor podľa nároku 1 alebo 2, v y z n a č u j ú c i s a t ý m, že každá pamäťová bunka (2;10; obr.4A, obr.4E) je schopná zahrnúť všetky informácie, ktoré sú nevyhnutné na vykonanie redukčnej operácie.
4. Redukčný procesor podľa nároku 3, v y z n a č u j ú c i s a t ý m, že uvedená informácia o redukcii obsahuje tiež referenciu (VALUE/DES) na minimálne jednu z uvedených pamäťových buniek, pričom obsah pamäťových buniek je spojený uvedenou referenciou do stromovej štruktúry (obr.6B, 6G, 7B, 8B; obr.5A až 5F) .
5. Redukčný procesor podľa nárokov 1 až 4, v y z n a č u j ú c i s a t ý m, že minimálne jedna z uvedených pamäťových buniek, ktorá sa nazýva jadrová bunka (2), je schopná vykonávať všetky druhy redukcíí a zvyšok uvedených buniek, ktoré sa

nazývajú pamäťové bunky (10; obr.4A), sú schopné vykonávať iba určité časti niektorých druhov redukcií.

6. Redukčný procesor podľa nároku 5, v y z n a č u j ú c i s a t ý m, že uvedené bunky pamäťového ukladania sú obsiahnuté v asociatívnej pamäti, ktorá má usporiadanú prvú zbernicu (any type, Vr, cpb, set.s, match, r/w.s, r/w.b, r/w.r, Wand.a, Wor.b, reset.b, mode.a, mode.a*, prech, ba, mode.b, grant.b, prio, atd.) pre externé riadenie a usporiadanú druhú zbernicu (t_1 , t_2 , id, env, v_0 , v_1 , v_2 , v_3) pre dáta vrátane niekoľkých uvedených buniek objektového ukladania (10) pre uloženie zloženej informácie a prostriedkov (16,17) v každej uvedenej bunke objektového ukladania pre uloženie minimálne jednej značky, kde uvedené značky udávajú minimálne stav zvolenia alebo nezvolenia pre uvedenú bunku objektového ukladania a prostriedkov pre vykonanie operácie vyhľadávania medzi uvedenými bunkami objektového ukladania k nastaveniu uvedených značiek a dekodér priority (11), ku ktorému sú všetky uvedené bunky objektového ukladania pripojené a ktorý vyberie jednu z niekoľkých uvedených buniek objektového ukladania.
7. Redukčný procesor podľa nároku 6, v y z n a č u j ú c i s a t ý m, že minimálne jedna globálna zbernica (12, 13, 14) je určená na vykonávanie logických operácií typu AND a OR medzi uvedenými pamäťovými bunkami a prostriedkami (17) v každej pamäťovej bunke pre komunikáciu s uvedenými zbernicami a pre riadenie uvedenej pamäťovej bunky pri účasti na aktuálnej logickej operácii.
8. Redukčný procesor podľa nároku 6 alebo 7, v y z n a č u j ú c i s a t ý m, že každá bunka objektového ukladania obsahuje niekoľko polí objektového ukladania (IDENTIFIER, ENVIRONMENT, VALUE/DES.₀, VALUE/DES.₁, VALUE/DES.₂, VALUE/DES.₃), kde každé pole objektového ukladania je schopné uložiť dátové slovo s názvom num word a s príznakom v tvare príznakového slova.
9. Redukčný procesor podľa nárokov 6 až 8, v y z n a č u j ú c i s a t ý m, že každá pamäťová bunka obsahuje minimálne jedno pole pre uloženie atribútu (LAZY, WHERE, TYPE), ktoré udáva stav alebo stavy obsahu uvedenej pamäťovej bunky.

10. Redukčný procesor podľa nárokov 5 až 9, v y z n a č u j ú c i s a t ý m, že uvedená pamäťová bunka je aritmetickou jednotkou pre štruktúralne aritmetické spracovanie, ktoré obsahuje:
- a. minimálne jeden prostriedok pre vstup/výstup (v_0 , v_1 , v_2 , v_3 , id , env) pre vstup a výstup zoznamu dát do a z uvedenej bunky objektového ukladania,
 - b. niekoľko registrov ($S_{0.0}$ až $S_{3.3}$, F_0 až F_3 , ID , ENV), kde každý je prispôsobený na uloženie dátového slova, každé dátové slovo má časť príznaku, príznakové slovo a časť informácie, $num\ word$, uvedená časť príznaku obsahuje príznak, ktorý udáva, či daný register je v činnosti alebo nie, každý uvedený zoznam je uložitelný v dopredu definovanom počte uvedených registrov, uvedená časť príznaku každého uvedeného registra medzi uvedenými registrami je označená ako používaná a udáva, či jeden z uvedených zoznamov má uloženú minimálne časť v aktuálnom registri a že uvedený zoznam, ktorý má uloženú časť v uvedenom aktuálnom registri, obsahuje zoznam inštrukcií, akého druhu tento zoznam je a kde je zřejmý vzťah medzi uvedenými zoznamami z usporiadania uvedených zoznamov v uvedených registroch,
 - c. prostriedky riadenia (6) pre riadenie uvedených registrov a pre využitie uvedeného zoznamu inštrukcií, ktorý patrí zoznamom, ktoré sú uložené v uvedených registroch, aby sa preskupili uvedené zoznamy medzi uvedenými registrami a pre vstup/výstup obsahu registra v súlade s uvedeným zoznamom inštrukcií.
11. Redukčný procesor podľa nároku 10, v y z n a č u j ú c i s a t ý m, že uvedené zoznamy, ktoré sú uložené v uvedených registroch, sú usporiadané ako stromový zoznam, z ktorých jeden zo zoznamov je koreňový zoznam.
12. Redukčný procesor podľa nároku 11, v y z n a č u j ú c i s a t ý m, že má minimálne jeden register navyše (ID), v ktorom je možné uložiť identifikátor uloženého stromu zoznamu.
13. Redukčný procesor podľa nároku 11 alebo 12, v y z n a č u j ú c i s a t ý m, že má minimálne jeden register navyše (ENV), v ktorom je možné uložiť prostredie uloženého stromu zoznamu.

- 14.Redukčný procesor podľa nárokov 10 až 13, v y z n a č u j ú c i s a t ý m, že obsahuje maticu registrov ($S_{0,0}$ až $S_{3,3}$), ktorá má periférnu radu ($S_{0,0}$ až $S_{3,3}$), vytvárajúcu hlavné registre, stĺpce uvedenej matice vytvárajú bázové registre.
- 15.Redukčný procesor podľa nároku 14, v y z n a č u j ú c i s a t ý m, že obsahuje niekoľko pomocných registrov (F0 až F3), ktoré sú mimo uvedenej matice.
- 16.Redukčný procesor podľa nárokov 10 až 15, v y z n a č u j ú c i s a t ý m, že uvedený koreňový zoznam stromu zoznamov je prispôbený na tomu, aby bol umiestnený v rôznych registroch v závislosti na úrovni uvedeného aktuálneho stromu, ktorý má byť uložený.
- 17.Redukčný procesor podľa nárokov 10 až 16, v y z n a č u j ú c i s a t ý m, že informácia, týkajúca sa druhu redukcie, ktorá sa má vykonať, môže byť odvodená od typu identifikátora, prostredia a stromu hodnôt, kde uvedený identifikátor, prostredie a každá individuálna hodnota je prvok v uvedenom spínači.
- 22.Redukčný procesor podľa nárokov 8 až 21, v y z n a č u j ú c i s a t ý m, že uvedený strom hodnôt obsiahnutý v uvedenom spínači obsahuje listové prvky, teda koncové prvky a zložené prvky, kde každý zložený prvok obsahuje stav a zoznam hodnôt.
- 23.Redukčný procesor podľa nárokov 8 až 22, v y z n a č u j ú c i s a t ý m, že uvedený zápis vykonávateľnosti v uvedenom spínači má minimálne dva stavy, z ktorých prvý stav je kľudový stav a druhý stav je stav vykonávania.
- 24.Redukčný procesor podľa nárokov 8 až 22, v y z n a č u j ú c i s a t ý m, že uvedený zápis polohy v uvedenom spínači má minimálne dva stavy, z ktorých prvý stav je poloha uzla a druhý stav je poloha koreňa.
- 25.Redukčný procesor podľa nárokov 8 až 24, v y z n a č u j ú c i s a t ý m, že každý prvok v uvedenom spínači je zložený z príznakového slova a num word, uvedené num word je zložené z niekoľkých bitov, kde každý má hodnotu buď true alebo false.
- 26.Redukčný procesor podľa nárokov 12 až 25, v y z n a č u j ú c i s a t ý m, že uvedené príznakové slová sú rozdelené na nepriamu triedu a priamu triedu.

- 27.Redukčný procesor podľa nárokov 8 až 26, v y z n a č u j ú c i s a t ý m, že je prispôsobený na použitie prvého kódovania binárneho slova, ktoré predstavuje typ integer a druhé kódovanie binárneho slova, ktoré predstavuje typ s plávajúcou rádovou čiarkou, a tým, že uvedené kódovania sú také, že reprezentácia s plávajúcou rádovou čiarkou je zaistená rovnakým spôsobom, ako reprezentácia typu integer.
- 28.Redukčný procesor podľa nárokov 8 až 27, v y z n a č u j ú c i s a t ý m, že typ integer je prispôsobený na uloženie v uvedenom prvku v binárne kódovanom formáte tak, že všetky hodnoty sú uvedené od najmenšej po najväčšiu hodnotu v rade čísel, kde každé číslo obsahuje niekoľko bitových hodnôt a v ktorých je nula reprezentovaná v strede uvedenej rady čísel a s binárnymi číslami, ktoré majú binárne hodnoty true na mieste najvýznamnejšieho bitu a ktoré majú vo zvyšku svojich bitov binárnu hodnotu false.
- 29.Redukčný procesor podľa nárokov 8 až 28, v y z n a č u j ú c i s a t ý m, že binárne kódovaná hodnota, ktorá je v plávajúcej rádovej čiarky, obsahuje znamienko, znamienko exponentu a kódové pole, pole exponentu a mantisy, uvedené znamienko exponentu a kódového pola obsahujú údaj o polohe rozdelenia medzi uvedeným polom exponentu a uvedeným polom mantisy tak, že pole exponentu a mantisy majú premennú dĺžku.
- 30.Redukčný procesor podľa nároku 29, v y z n a č u j ú c i s a t ý m, že uvedené slová, ktoré predstavujú číselné hodnoty, majú kódovania, ktoré vykonávajú uvedené číselné hodnoty do reprezentácie hustoty, teda že každá kódovaná reprezentácia hodnoty zodpovedá iba jednej interpretovanej hodnote.
- 31.Redukčný procesor podľa nárokov 8 až 30, v y z n a č u j ú c i s a t ý m, že minimálne niektoré z uvedených zápisov spínača sú uvedené v rôznych stavoch, pričom každý je zložený z niekoľkých bitov, ktoré predstavujú binárny kód.
- 32.Redukčný procesor podľa niektorého z predchádzajúcich nárokov, v y z n a č u j ú c i s a t ý m, že používa číslicovú aritmetickú jednotku na vykonanie aritmetických, logických a relačných operácií s prvkami číselných hodnôt, vrátane:

- a. vstupu, ktorý obsahuje súbor zberníc, z ktorých každá zbernica, ktorá je platná, je vybavená prvkom zoznamu v zozname, kde uvedený zoznam obsahuje informáciu o inštrukcii s ohľadom na slová v uvedenom zozname,
 - b. prostriedkov spracovania, ku ktorým uvedený súbor zberníc je pripojený a vykonáva operácie na uvedených slovách v uvedenom zozname s použitím uvedenej informácie o inštrukcii prepísaním uvedených slov v súlade s uvedenou inštrukciou,
 - c. výstupu, prispôsobeného k prezentovaniu prepísaného výsledku vrátane súboru zberníc, ktoré majú rovnaký počet a konfiguráciu ako uvedený vstupný súbor zberníc.
33. Redukčný procesor podľa nároku 32, v y z n a č u j ú c i s a t ý m, že minimálne jeden prvok zoznamu v uvedenom zozname je rezervovaný pre reprezentáciu informácie o inštrukcii a pre stanovenie na konkrétnom uvedenom súbore vstupných zberníc, a reprezentácie číselnej hodnoty, ktoré majú byť vypočítané, sú prispôsobené na to, aby boli odovzdané na ostatných vstupných zberniciach a že uvedené prostriedky spracovania sú prispôsobené na vykonanie výpočtu prepísaním reprezentácie číslícovej hodnoty v uvedenom vstupnom súbore.
34. Redukčný procesor podľa nároku 32, v y z n a č u j ú c i s a t ý m, že niekoľko obvodov, kde každý je prispôsobený na vykonávanie špecifickej operácie na uvedenej reprezentácii číselnej hodnoty na uvedených vstupných zberniciach, je určených k paralelnému odovzdaniu svojich výsledkov uvedených operácií a že prostriedky riadenia (1a, 27), ktoré sú určené pre uvedenú informáciu o inštrukcii sú prispôsobené na výber vykonaných výsledkov medzi všetkými vykonanými výsledkami adekvátne pre vykonanie aktuálnej inštrukcie.
35. Redukčný procesor podľa nároku 34, v y z n a č u j ú c i s a t ý m, že keď je uvedený zoznam prispôsobený na to, aby obsahoval funkčnú aplikáciu, kde jeden z jeho prvkov je inštrukčný kód a zvyšok sú argumenty pre uvedenú inštrukciu, uvedené prostriedky pre spracovanie, ktoré vykonávajú inštrukciu prepisom a recykláciou uvedeného inštrukčného kódu z uvedeného výstupu na uvedený vstup, až sa dosiahne prípadný výsledok, že

pri každom uvedenom prepise sú uvedené prostriedky pre spracovanie prispôsobené na prepis uvedeného zoznamu tak, aby obsahoval revidované slovo inštrukčného kódu, pokiaľ to je príslušné pro aktuálny výpočet, po ktorom nasledujú slová hodnoty.

- 36.Redukčný procesor podľa ktoréhokoľvek z predchádzajúcich nárokov v y z n a č u j ú c i s a t ý m, že množina procesorov je prepojená navzájom sieťou, kde uvedené prepojené procesory prvého rádu predstavujú redukčný procesor druhého rádu.
- 37.Redukčný procesor podľa ktoréhokoľvek z predchádzajúcich nárokov v y z n a č u j ú c i s a t ý m, že jeden alebo niekoľko uvedených pamäťových buniek v každom redukčnom procesore prvého rádu je prispôsobených na uloženie spínača,čo je označenie vykonávateľnosti, polohy v strome stromovej štruktúry, identifikátora, prostredia a stromu hodnôt, uvedený identifikátor, prostredie a každá individuálna hodnota, ktorá je prvok v uvedenom spínači, každý prvok v uvedenom spínači je zložený z niekoľkých bitov, kde každý z nich má hodnotu true alebo false a uvedené príznakové slovo sú rozdelené na nepriamu triedu a priamu triedu a že bitový vzorec uvedeného prvku nepriamej triedy je rozdelený na oblasťnú časť a adresovú časť.
- 38.Redukčný procesor podľa nároku 37, v y z n a č u j ú c i s a t ý m, že niekoľko procesorov prvého rádu ($FOP_{1.1}$ až $FOP_{m.m}$) je prepojené do štvorcového poľa, ktoré predstavuje redukčný procesor druhého rádu a že každý procesor druhého rádu v uvedenom štvorcovom poli obsahuje kanál (CAN), ktorý ide ku každému susednému kanálu redukčného procesoru prvého rádu v uvedenom štvorcovom poli (obr.37A, 37B a 37D).
- 39.Redukčný procesor podľa nároku 38, v y z n a č u j ú c i s a t ý m, že uvedený redukčný procesor druhého rádu je rozdelený na logické oblasti (obr. 37C), kde každá má rozmer 2^n krát 2^n redukčných procesorov prvého rádu, pričom uvedené logické oblasti sú umiestnené po strane každého v pravidelnom vzorci,aby sa pokrylo uvedené štvorcové pole uvedeného redukčného procesoru prvého rádu.
- 40.Redukčný procesor podľa nároku 39, v y z n a č u j ú c i s a

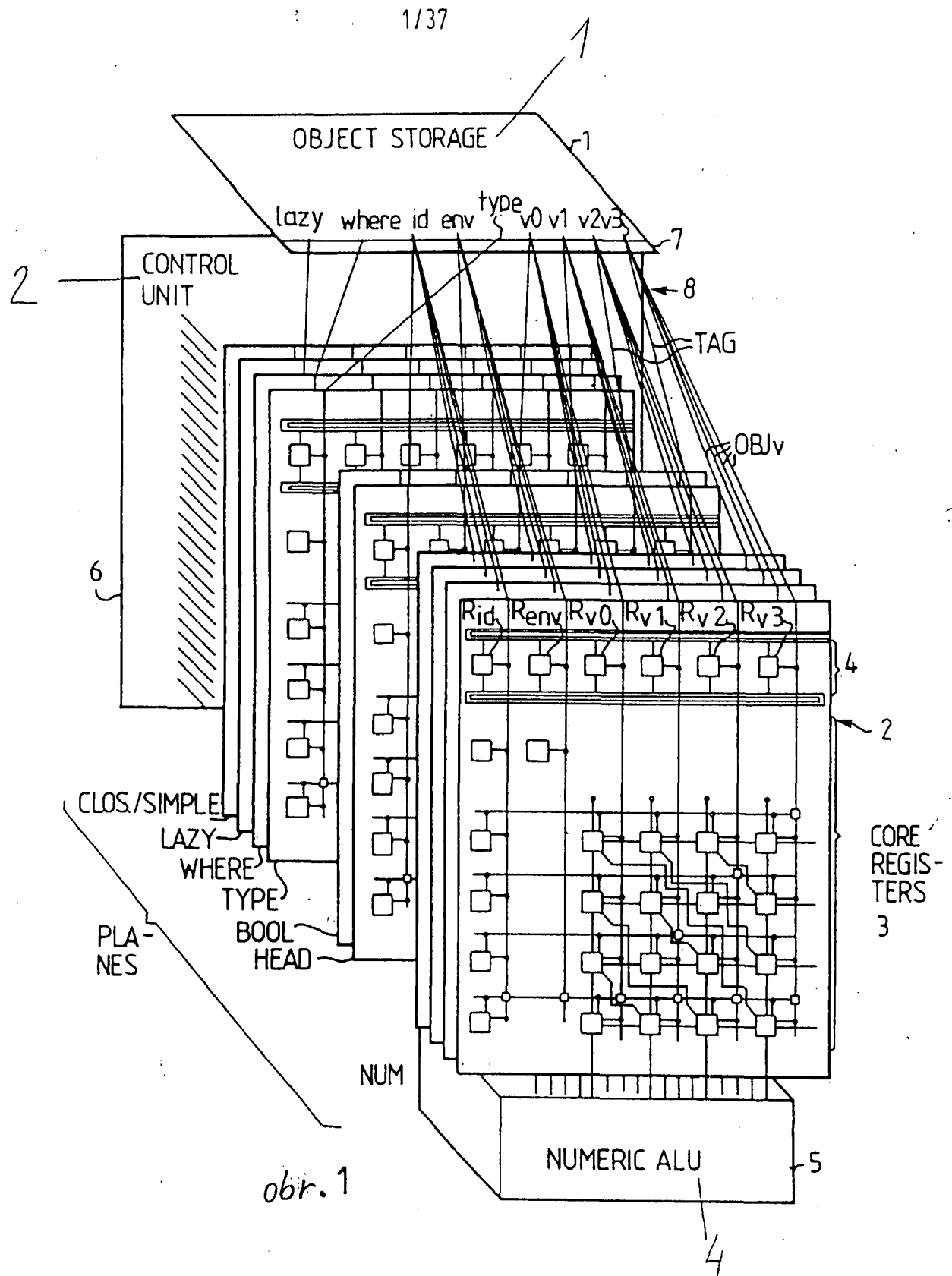
t ý m, že procesory prvého rádu sú pripojené navzájom v uvedenom procesore druhého rádu tak, že každá uvedená oblasť vo vnútri procesora druhého rádu je posunuteľná o polovicu oblasti v každom smere internou novou definíciou posunutia uvedenej oblasti vo vnútri uvedeného procesoru druhého rádu.

- 41.Redukčný procesor podľa nároku 39 alebo 40 v y z n a č u j ú c i s a t ý m, že minimálne jeden adresovateľný nepriamy prvok s adresou je uložený v každej oblasti, pričom uvedený prvok je možno adresovať iba z pamäťových buniek v oblasti, ku ktorej patrí.
- 42.Redukčný procesor podľa nárokov 37 až 41, v y z n a č u j ú c i s a t ý m, že niekoľko redukčných procesorov prvého rádu je prepojených v hierarchii sietí (NET_1 až NET_n), kde každá sieť je zbernica.
- 43.Redukčný procesor podľa nárokov 37 až 42, v y z n a č u j ú c i s a t ý m, že niekoľko redukčných procesorov prvého rádu je prepojených v hierarchii sietí, kde každá sieť je okruh (obr.39).
- 44.Redukčný procesor podľa nárokov 37 až 43, v y z n a č u j ú c i s a t ý m, že niekoľko redukčných procesorov prvého rádu je prepojených v hierarchii sietí, kde sú minimálne dva druhy sietí, prvý druh je zbernica, druhý druh je okruh a tretí druh je štvorcové pole (obr. 40).
- 45.Redukčný procesor podľa ktoréhokoľvek z predchádzajúcich nárokov v y z n a č u j ú c i s a t ý m, že obsahuje minimálne jeden prostriedok portu, ktorý je pripojený k uvedeným aktívnym prostriedkom ukladania a minimálne jeden prostriedok pre prostredie, ktorý je pripojený k minimálne jednému uvedenému prostriedku portu.
- 46.Redukčný procesor podľa nároku 45, v y z n a č u j ú c i s a t ý m, že obsahuje prostriedky pre porovnanie postupnosti signálu (obr.34 a 35), ktorá je na prostriedkoch portu a uloženú postupnosť v minimálne jednej pamäťovej bunke, uvedená uložená postupnosť má možné nedefinované prvky postupnosti (\$) v uvedenom aktívnom prostriedku ukladania a prostriedky (CU, 3) pre prepísanie uvedenej uloženej postupnosti do nothing teda do niečoho, čo predstavuje rozpor pokiaľ porovnanie dá jasný rozdiel

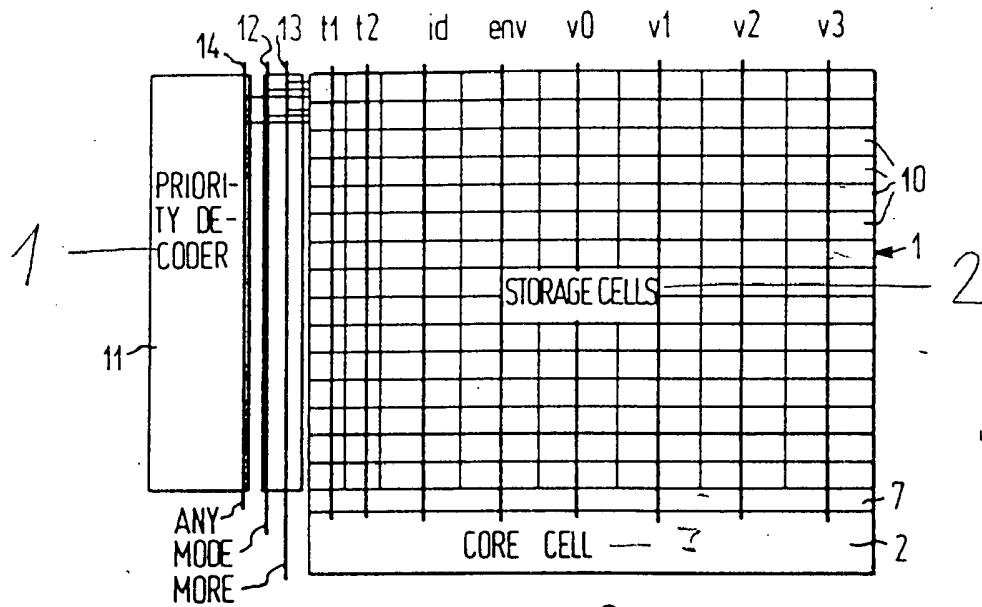
- alebo ináč pre prepísanie uvedenej uloženej postupnosti do špecifikovanej postupnosti, ktorá je unifikáciou uvedenej postupnosti signálu a uloženej postupnosti.
- 47.Redukčný procesor podľa nároku 46, v y z n a č u j ú c i s a t ý m, že uvedené prostriedky pre porovnanie vykonávajú porovnanie skupín dopredu určených čísiel zoznamu prvkov.
 - 48.Redukčný procesor podľa nárokov 46 alebo 47, v y z n a č u - c i s a t ý m, že obsahuje prostriedky (3, during, 7, 8, 9) pre privedenie uvedenej postupnosti signálu ako vzorkovaného signálu (obr. 34), ktorý sa mení s časom, ktorý má individuálne vzorkovacie periódy, uvedená postupnosť signálu je zoznam skupín prvkov, každá skupina obsahuje dobu trvania a minimálne jednu hodnotu signálu v priebehu tejto doby.
 - 49.Redukčný procesor podľa nároku 48, v y z n a č u j ú c i s a t ý m, že uvedený dopredu určený počet prvkov zoznamu v každej skupine je dva, usporiadané sú v páre, pričom každý pár obsahuje kombináciu času a hodnoty signálu.
 - 50.Redukčný procesor podľa nárokov 45 až 49, v y z n a č u j ú c i s a t ý m, že uvedené bunky objektového ukladania sú usporiadané v štruktúre v uvedenom aktívnom ukladaní, ktoré je prispôbené pre uloženie programu počítača v tvare explicitného alebo implicitného kódovania abstraktnej syntaxie, kde uvedená syntaxia popisuje niekoľko rôznych abstraktných objektov pomocou výrazov, každý prostriedok bunky objektového ukladania je schopný v danom okamžiku uložiť minimálne časť jedného uvedeného výrazu syntaxe v tvare vhodných dát a/ alebo programovej štruktúry.
 - 51.Redukčný procesor podľa nároku 44, v y z n a č u j ú c i s a t ý m, že množina geograficky samostatných navzájom prepojených redukčných procesorov druhého rádu (GSOP) tvorí redukčný procesor tretieho rádu.
 - 52.Redukčný procesor podľa nároku 51, v y z n a č u j ú c i s a t ý m, že asociatívne adresovanie, obmedzené podľa oblastí, je prispôbené pre redukčné procesory druhého rádu (SOP) a fyzické adresovanie je prispôbené pre redukčné procesory tretieho rádu.
 - 53.Redukčný procesor podľa nároku 52, v y z n a č u j ú c i s a

t ý m, že každý geograficky samostatný redukčný procesor druhého rádu (GSOP) v redukčnom procesore tretieho rádu obsahuje prostriedky pre sledovanie otcov, uložených v iných procesoroch GSOP ako je syn.

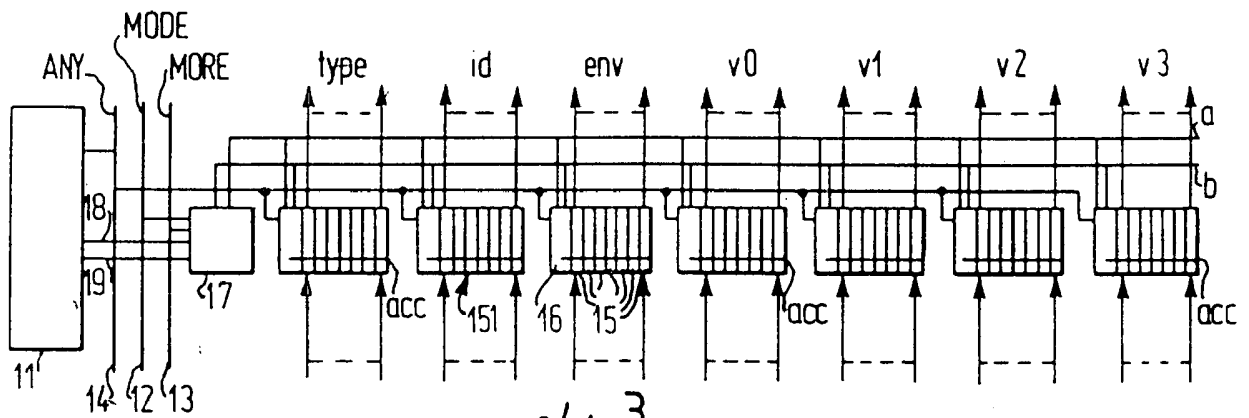
54. Redukčný procesor podľa nároku 53, v y z n a č u j ú c i s a t ý m, že uvedené prostriedky pre sledovanie uvedených nelokálnych otcov obsahujú dátové štruktúry v lokálnom objektovom ukladaní, uvedené dátové štruktúry obsahujú zoznam pre každého syna, ktorý má nelokálneho otca, uvedený zoznam obsahuje nelokálneho otca daného syna, uvedená adresa obsahuje časť, ktorá identifikuje procesor druhého rádu (GSOP), v ktorom je nelokálny otec uložený.



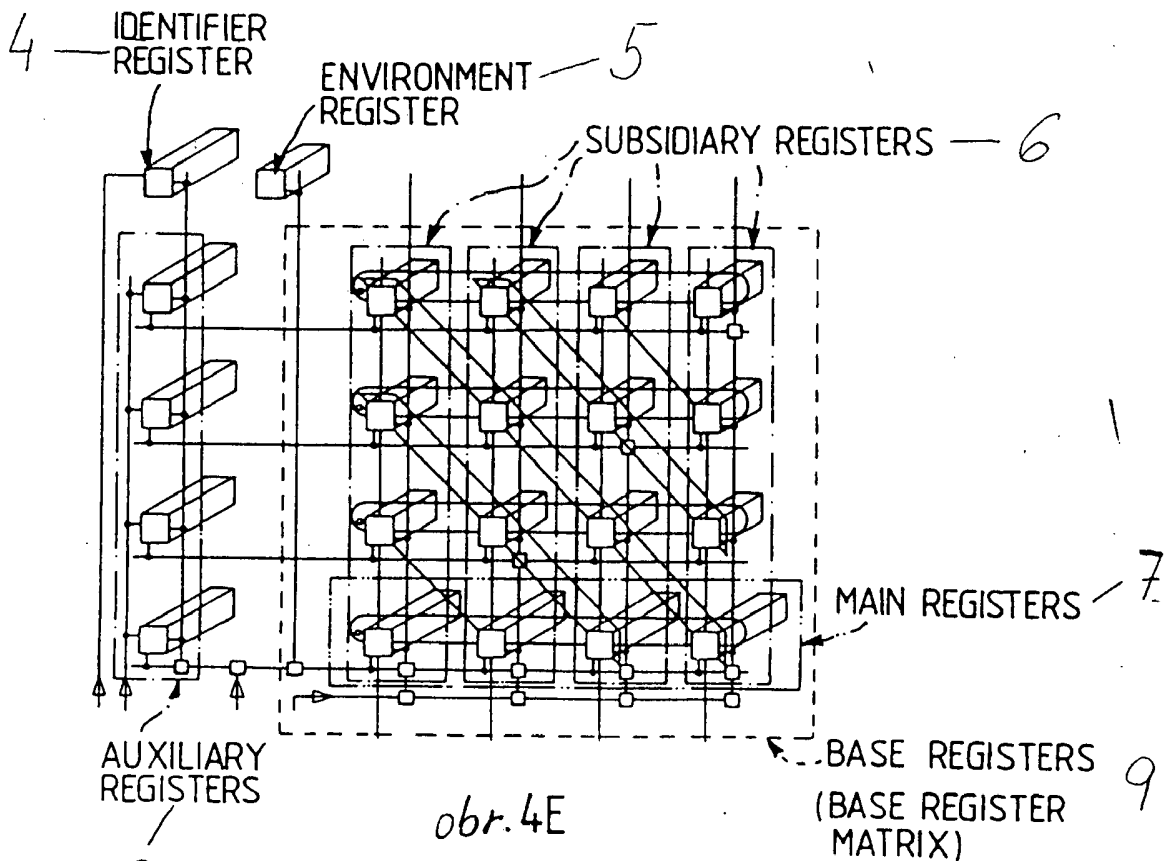
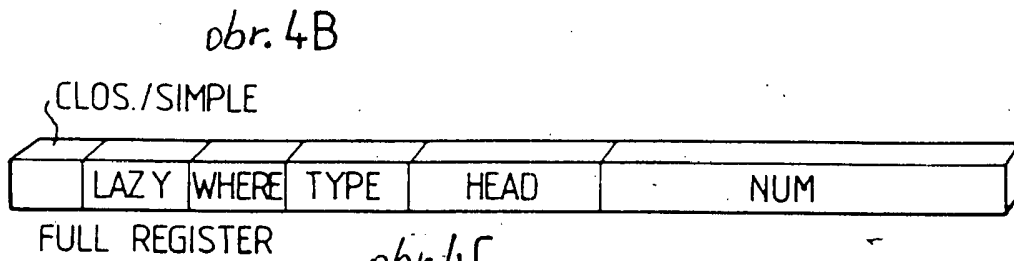
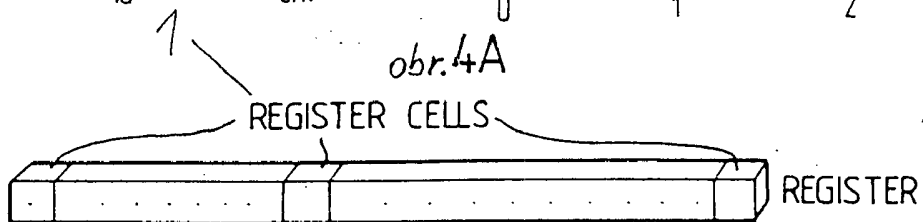
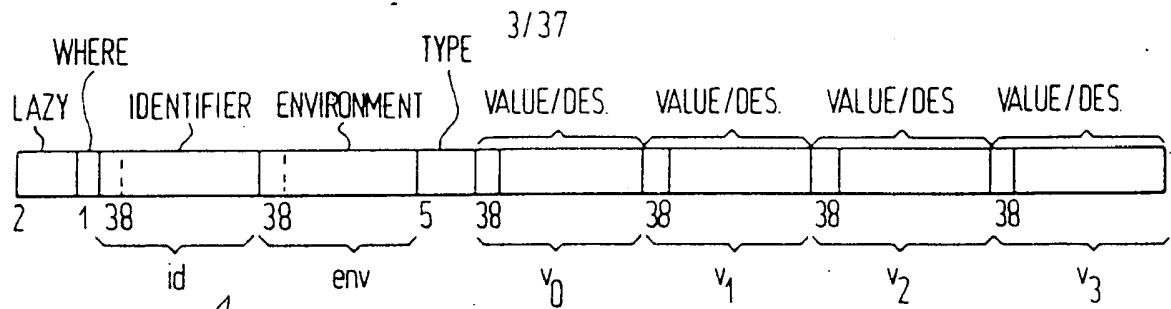
2/37



obr. 2

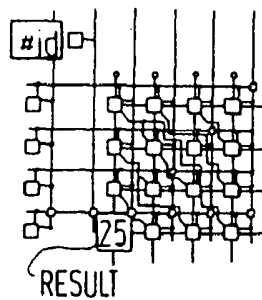


obr. 3



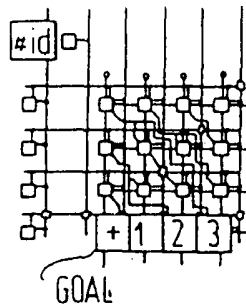
4/37

1
SIMPLE VALUE OR
MACHINE IDENTIFIER



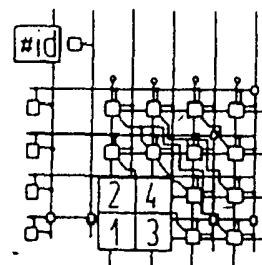
obr. 5A

2
ONE LEVEL STRUCTURE



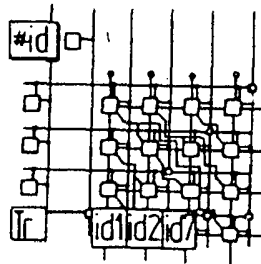
obr. 5B

3
TWO LEVEL STRUCTURE



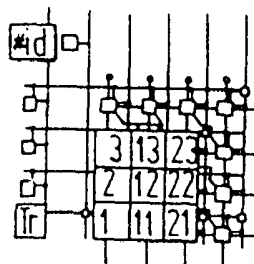
obr. 5C

3
TWO LEVEL STRUCTURE



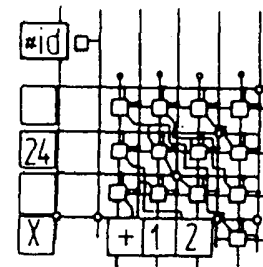
obr. 5D

4
THREE LEVEL STRUCTURE



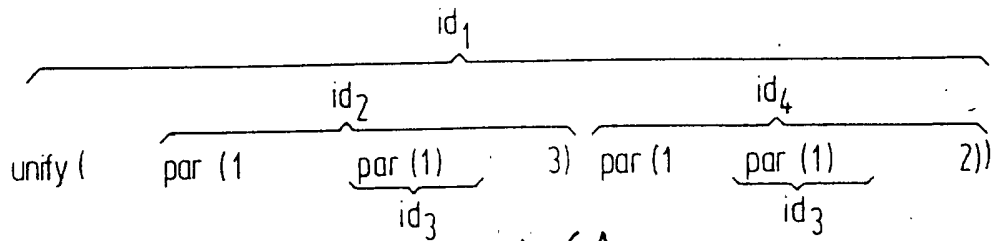
obr. 5E

5
IN PIPELINE MODE



obr. 5F

5/37



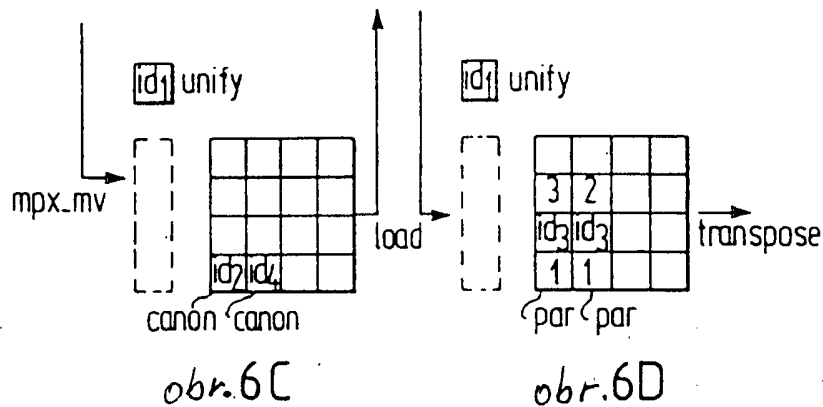
obr. 6A

exec	0	cls	id ₁			uni- fy	ca- non	id ₂	ca- non	id ₄	unu- sed		unu- sed	
idle	0	ca- non	id ₂			par	discr	1	ca- non	id ₃	discr	3	unu- sed	
idle	0	ca- non	id ₃			par	discr	1	unu- sed		unu- sed		unu- sed	
idle	0	ca- non	id ₄			par	discr	1	ca- non	id ₃	discr	2	unu- sed	

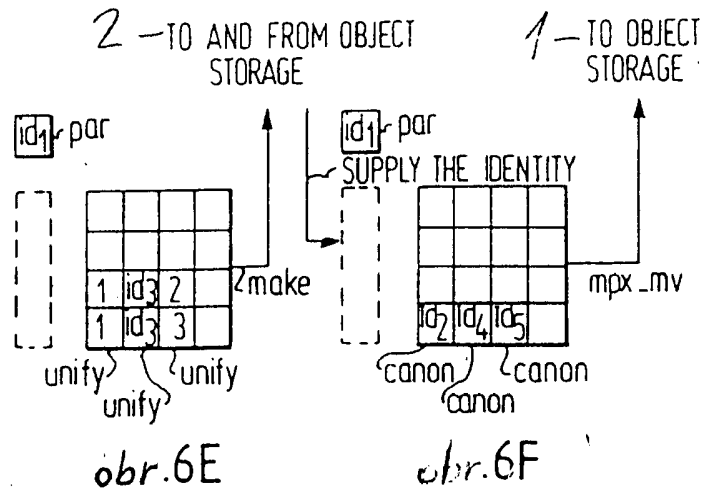
obr. 6B

1- FROM OBJECT STORAGE

2- TO & FROM OBJECT STORAGE

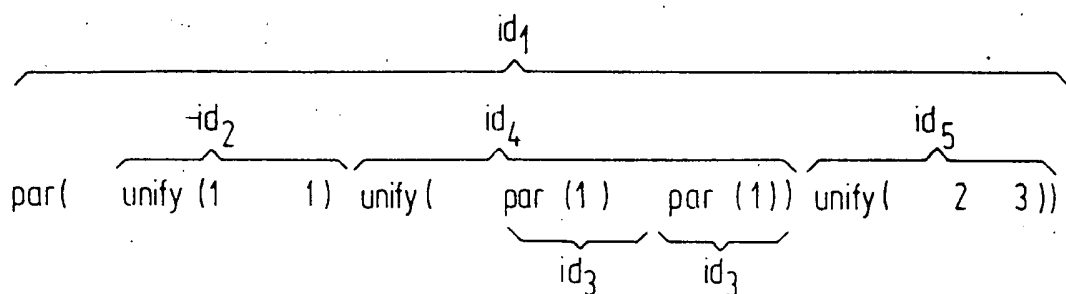


6/37

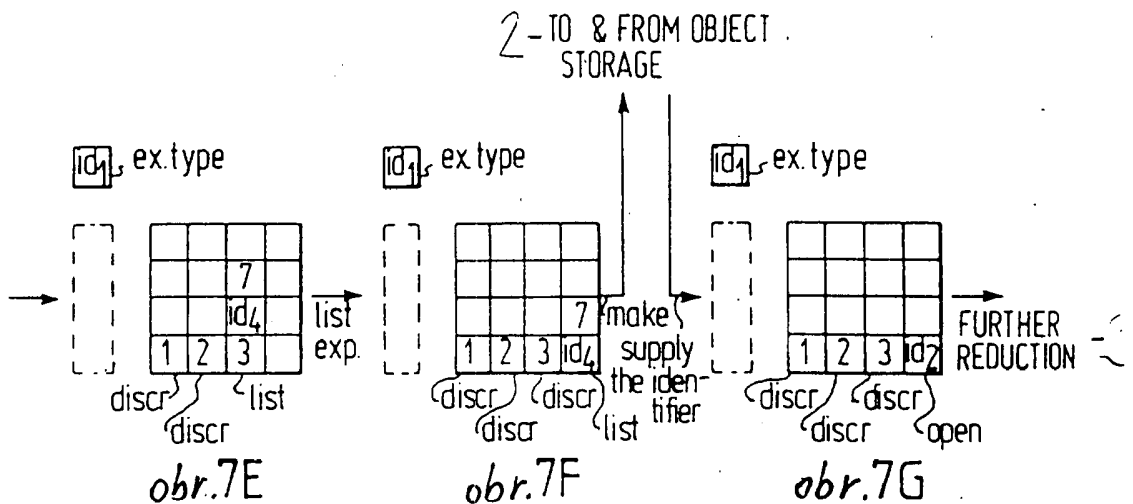
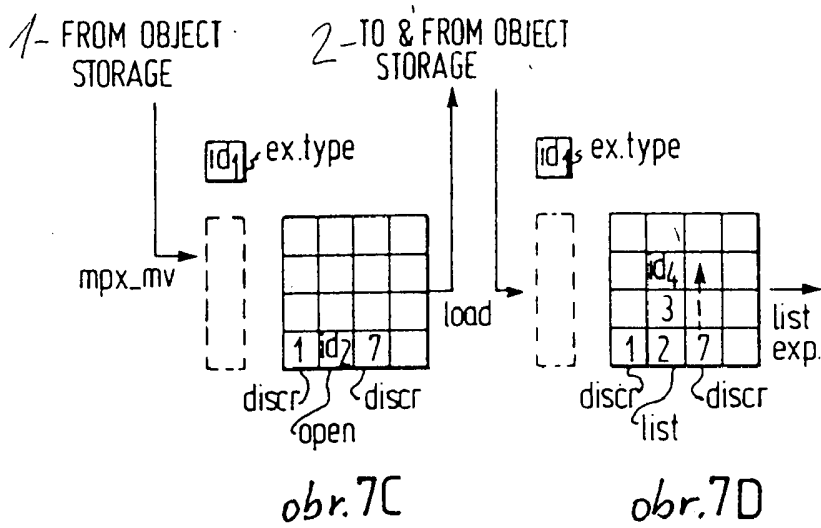
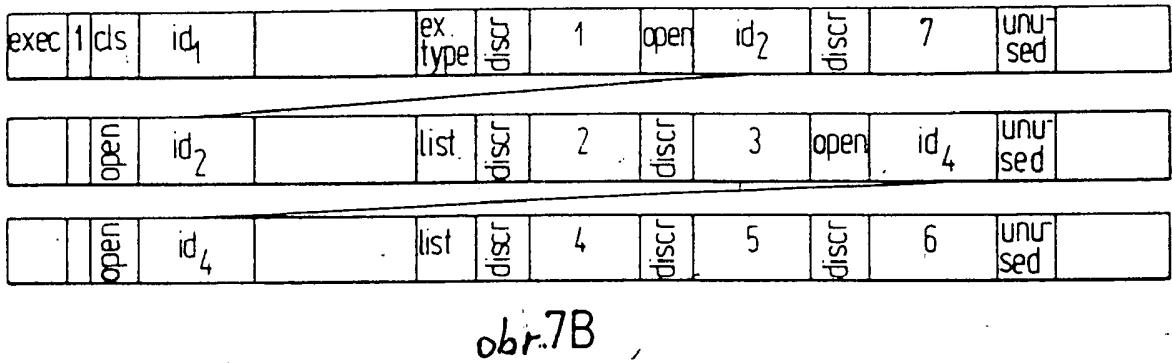
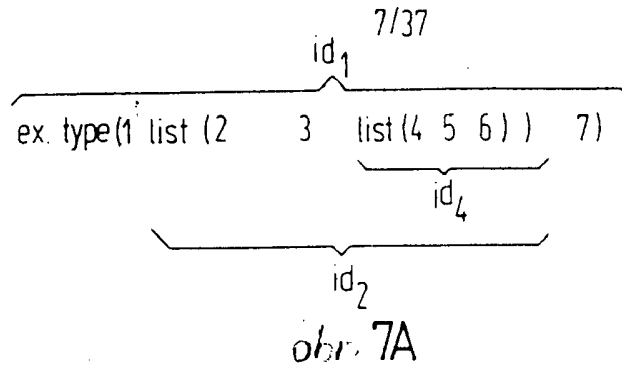


wait	-	cls	id ₁		par	cls	id ₂	cls	id ₄	cls	id ₅	un-	sed	
exec	-	cls	id ₂		uni-	ty	discr	1	discr	1	un-	sed		un-
-	-	ca-	non	id ₃		par	discr	1	un-	sed		un-	sed	
exec	-	cls	id ₄		uni-	ty	ca-	non	id ₃	ca-	non	id ₃	un-	sed
exec	-	cls	id ₅		uni-	ty	discr	2	discr	3	un-	sed		un-

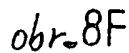
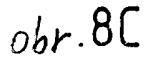
obr.6G



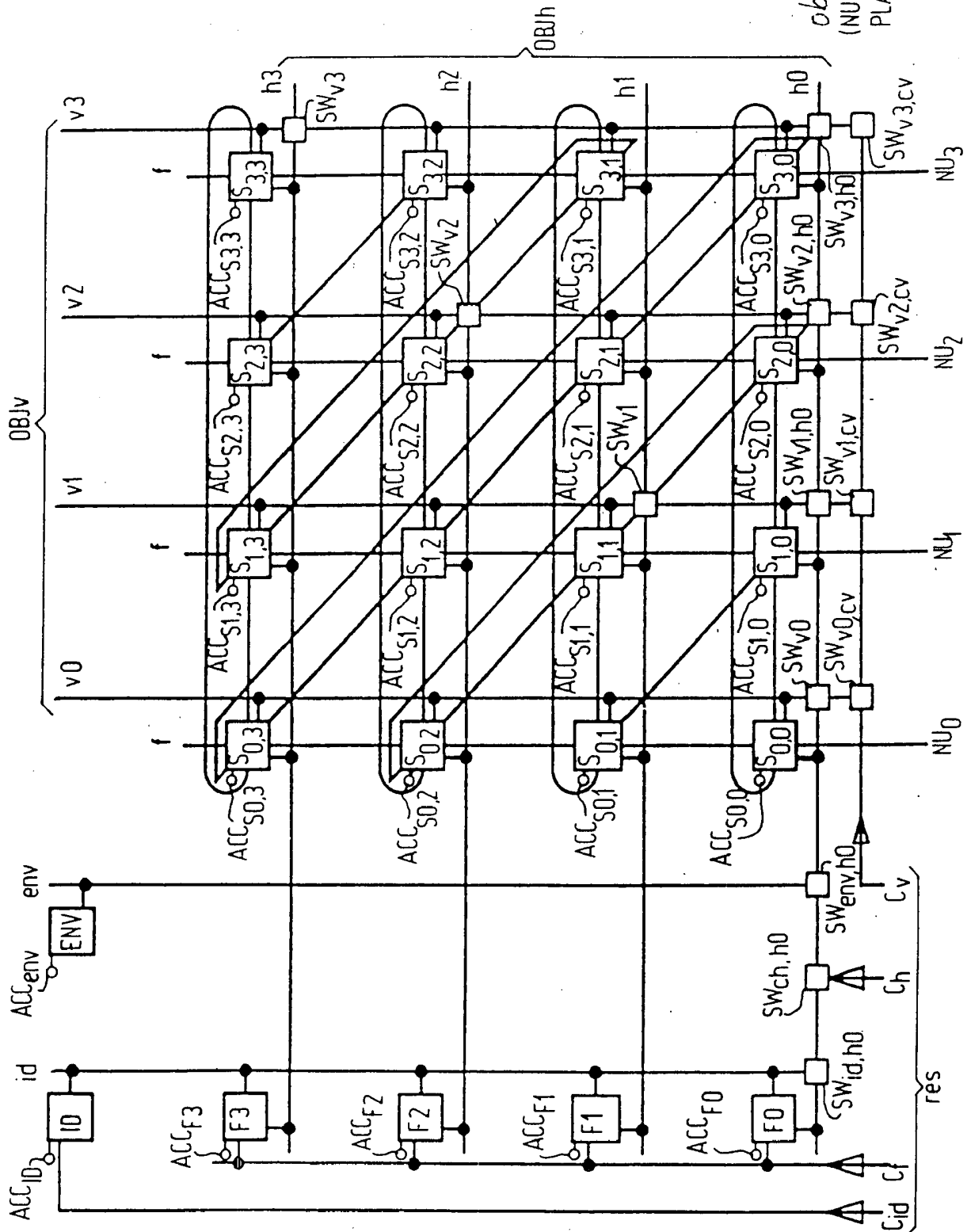
obr.6H



obr. 8A

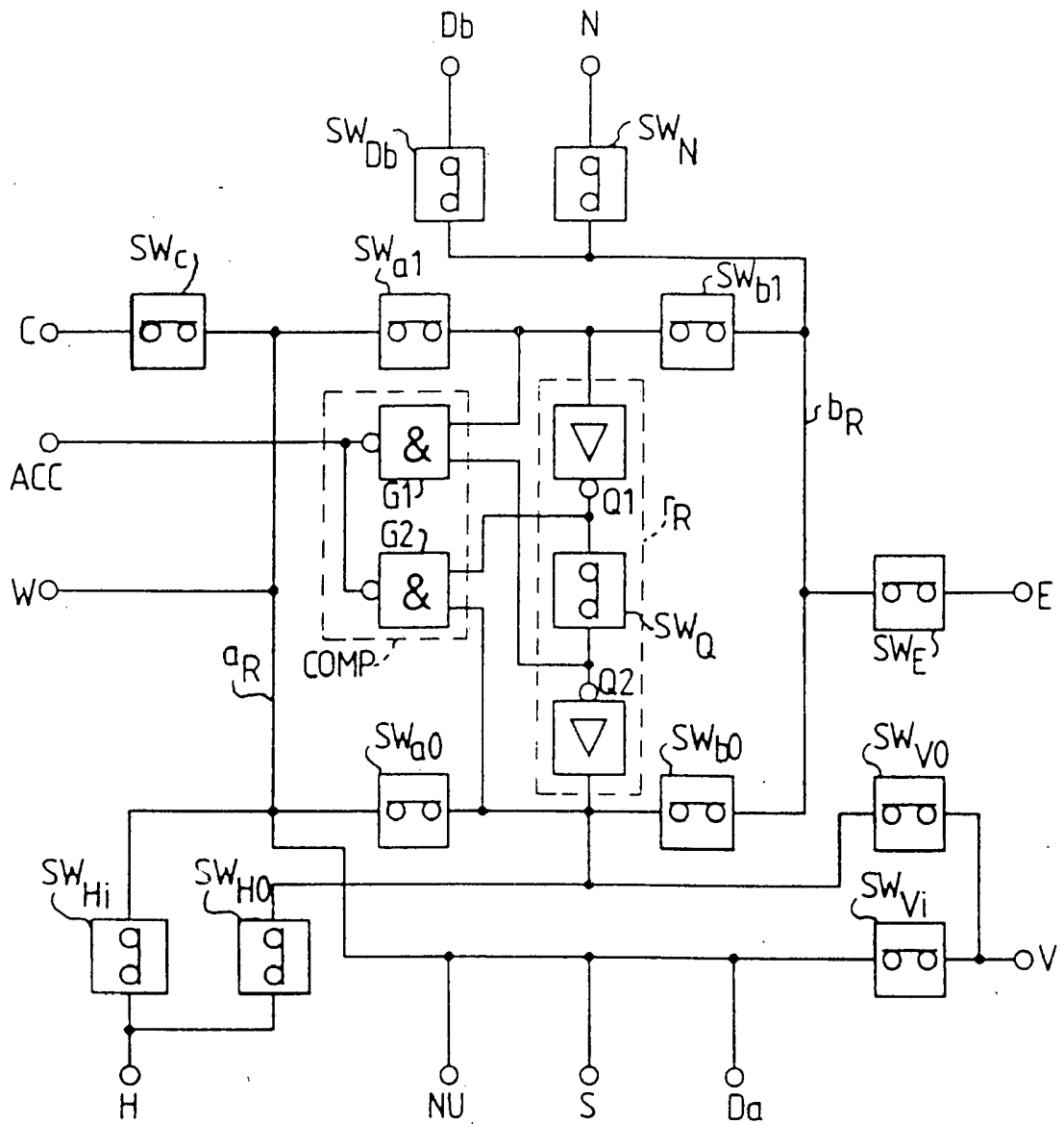
obr.8B

obr. 86

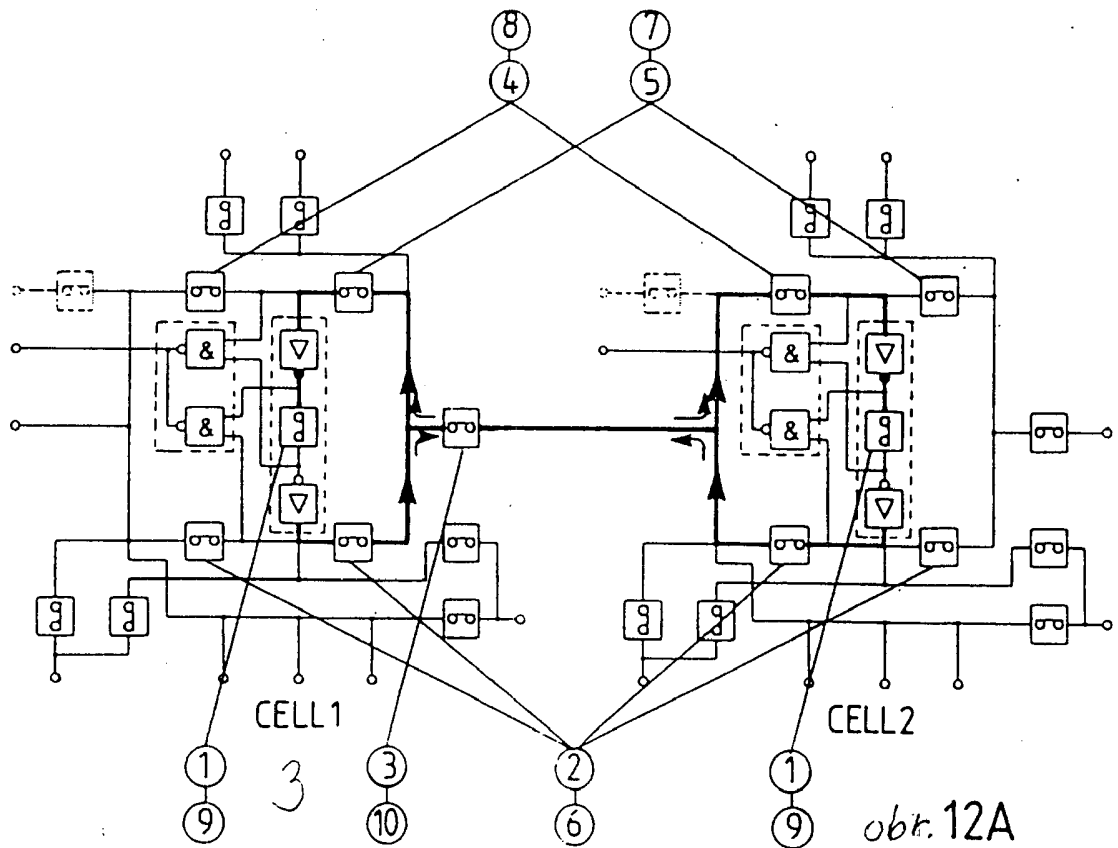
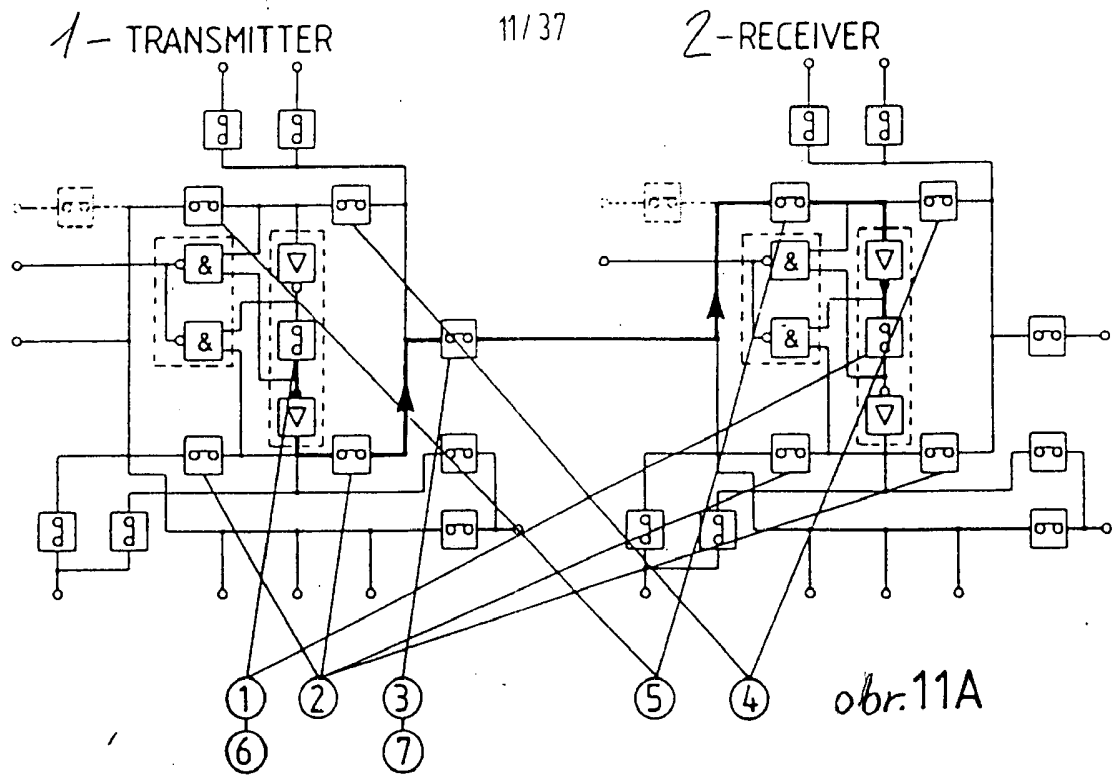


obr. 9
(NUM & HEAD
PLANES) 1

10/37

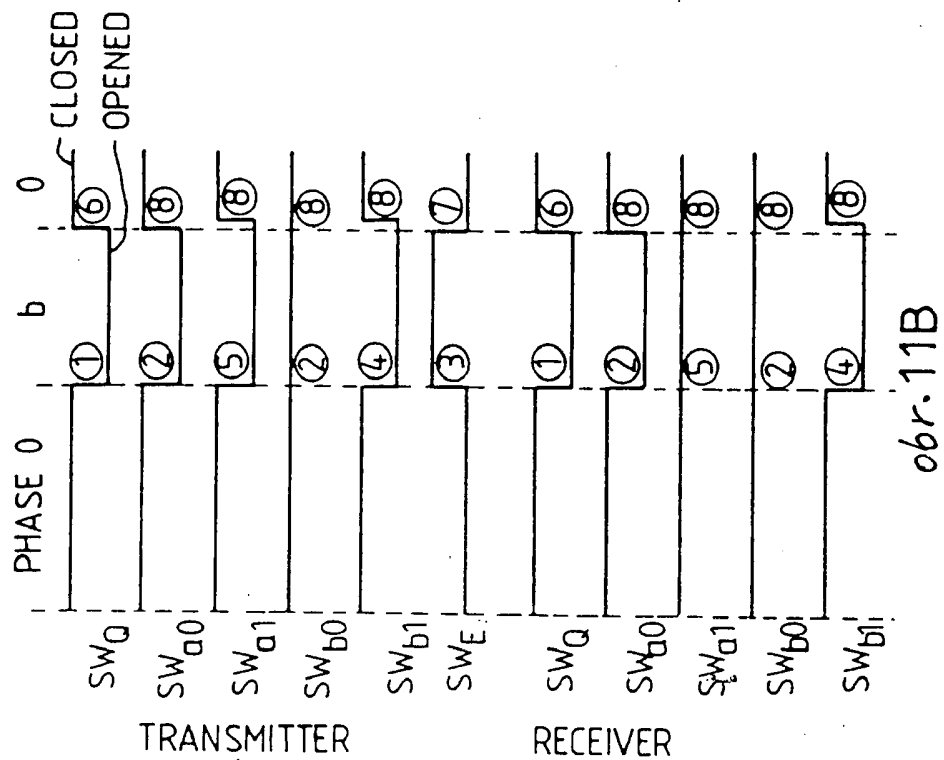
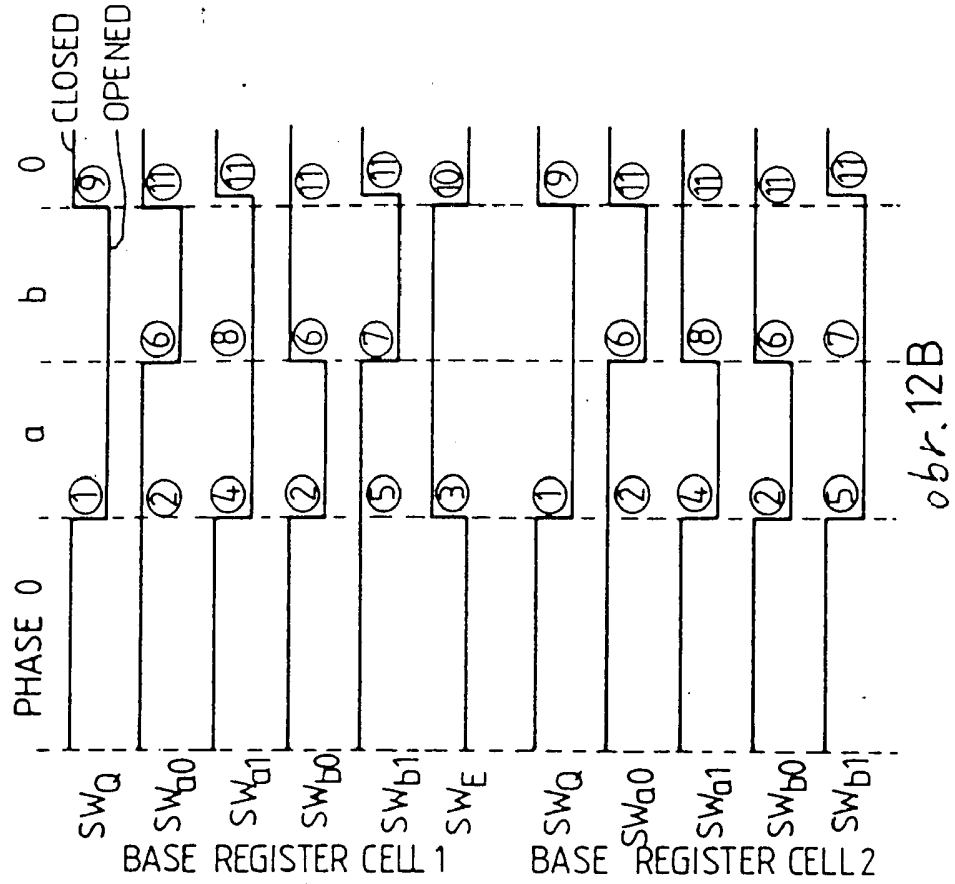


obr. 10
 1 GENERIC REGISTER CELL (NUM & HEAD PLANES)

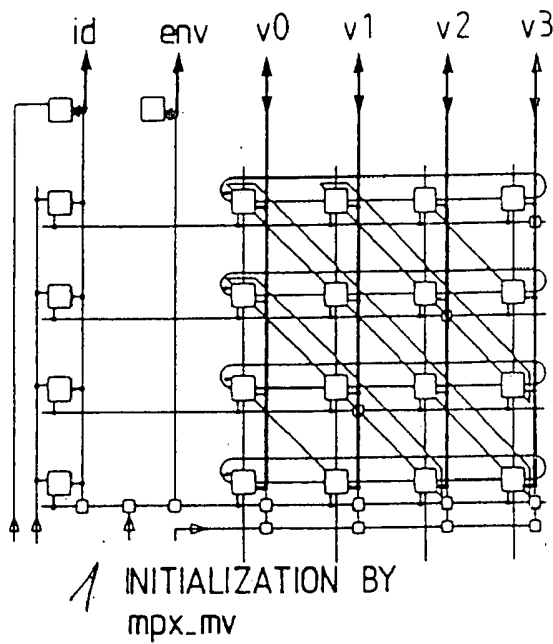


12/37

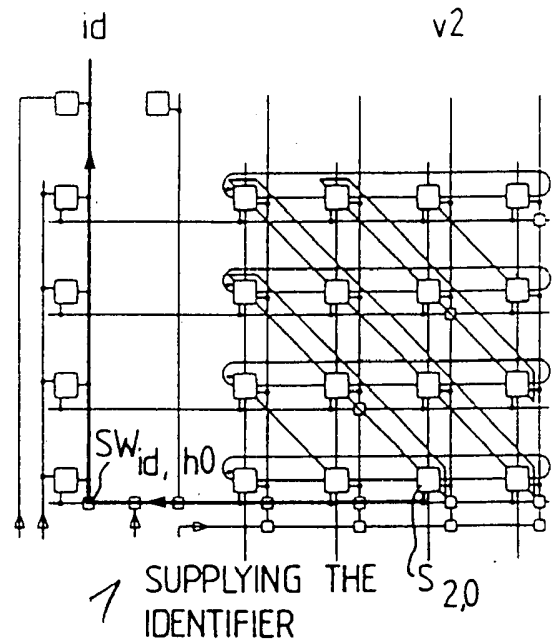
2 3



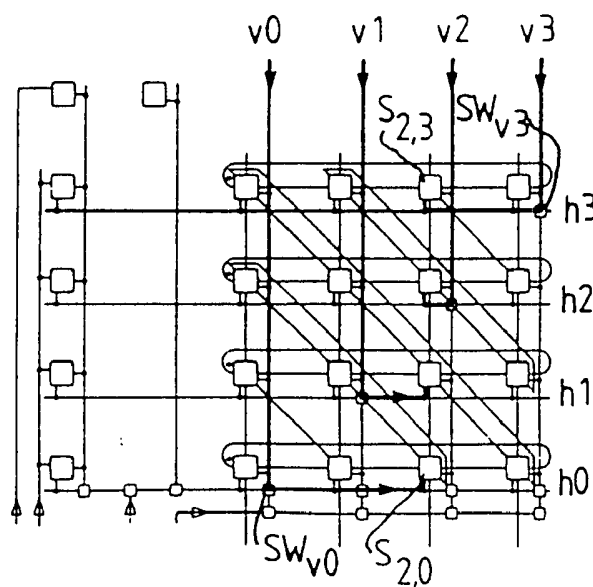
13/37



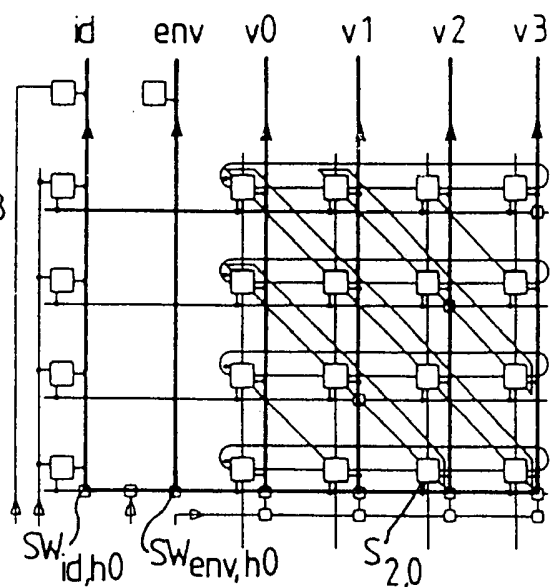
obr.13



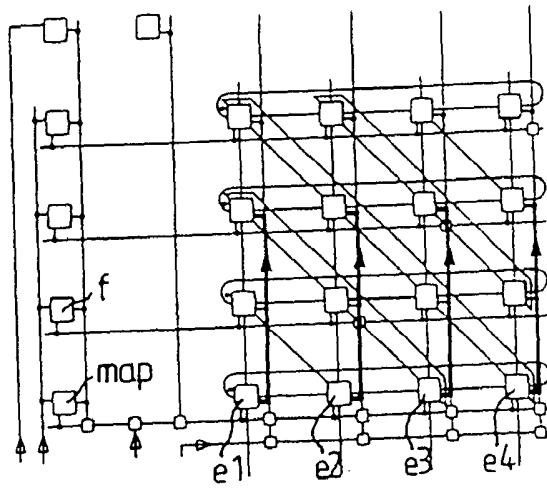
obr.14



obr.15

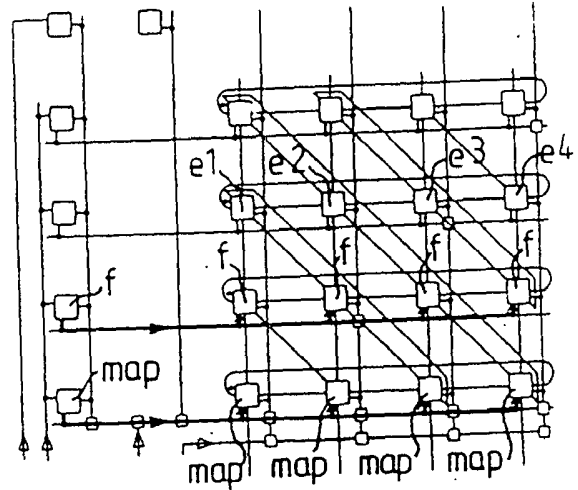


obr.16



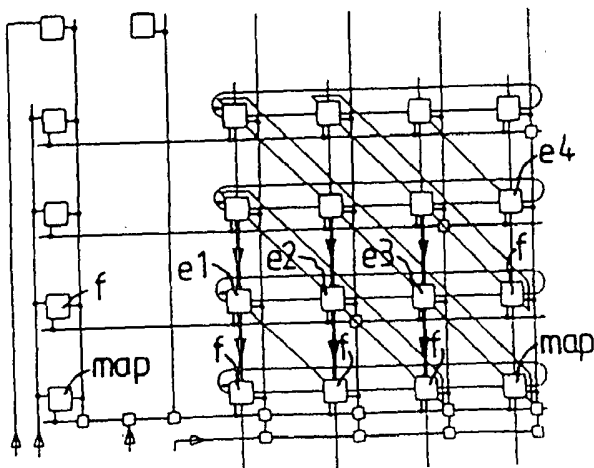
$(\text{map } f (e1 \ e2 \ e3 \ e4))$

obr. 17A



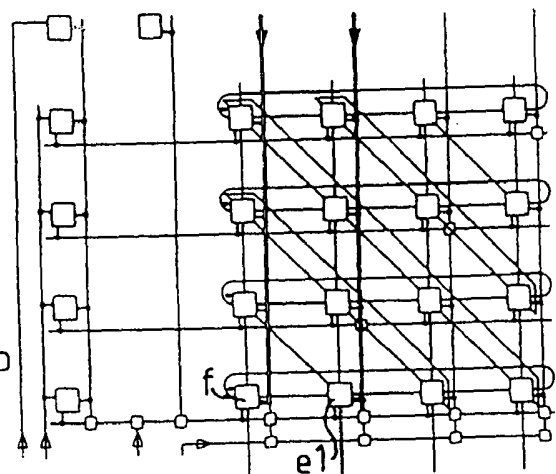
$((\text{map } f \ e1)(\text{map } f \ e2))$
 $(\text{map } f \ e3)(\text{map } f \ e4))$

obr. 17B



$((f \ e1)(f \ e2)(f \ e3))$
 $(\text{map } f \ e4))$

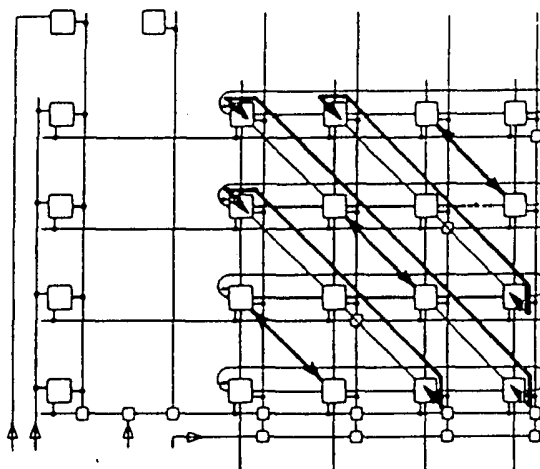
obr. 17C



$(f \ e1)$ RELOADED ↗

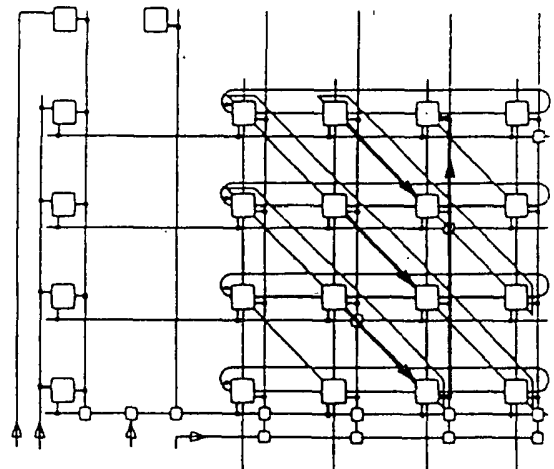
obr. 17D

15/37



TRANSPOSE

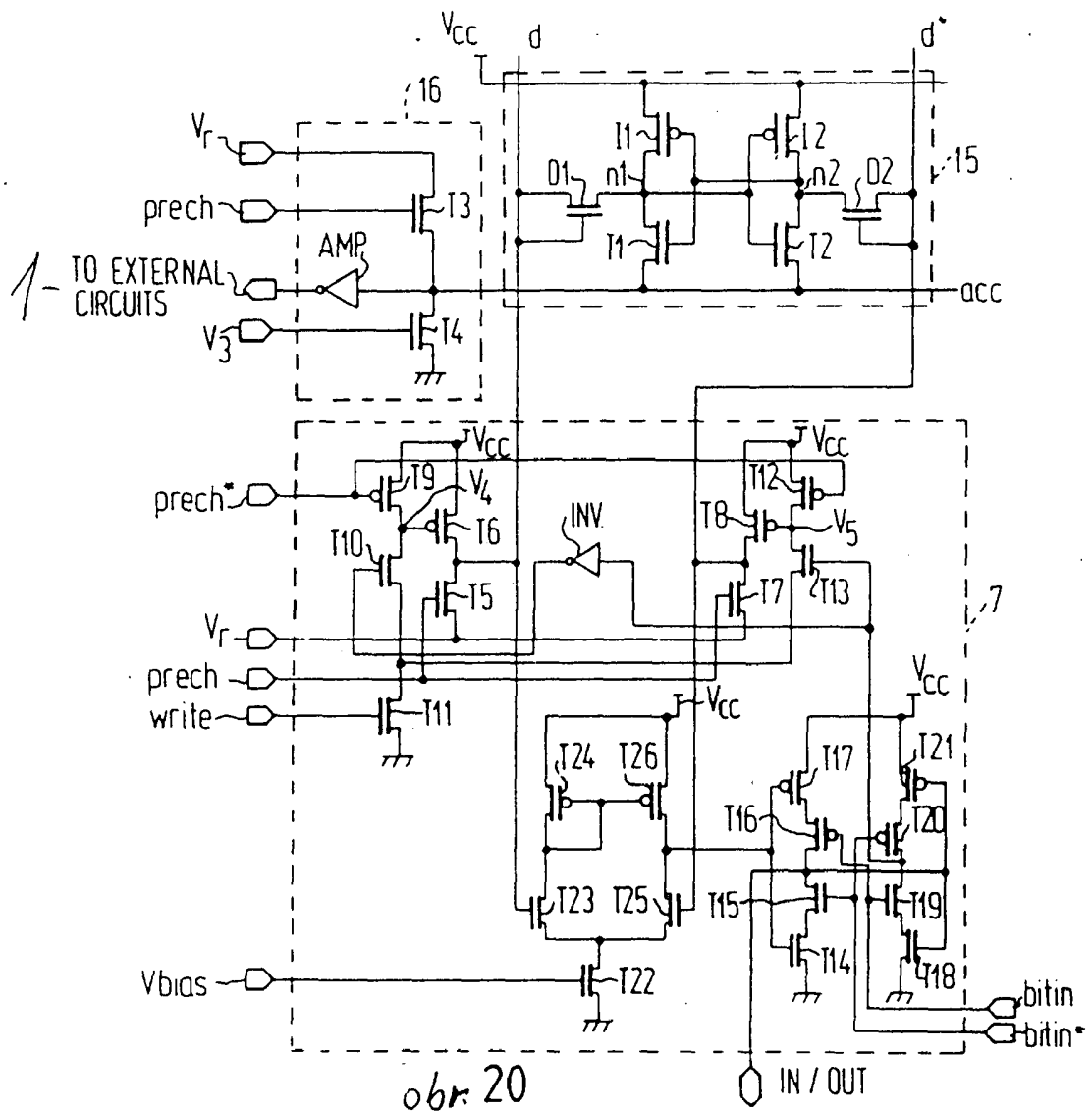
obr.18

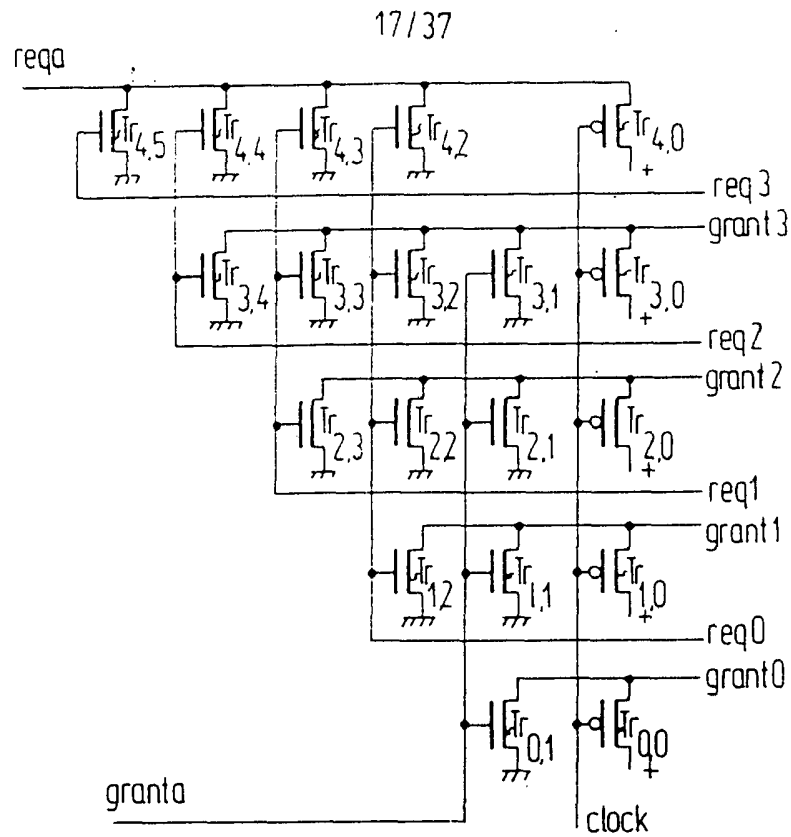


EXPAND_list

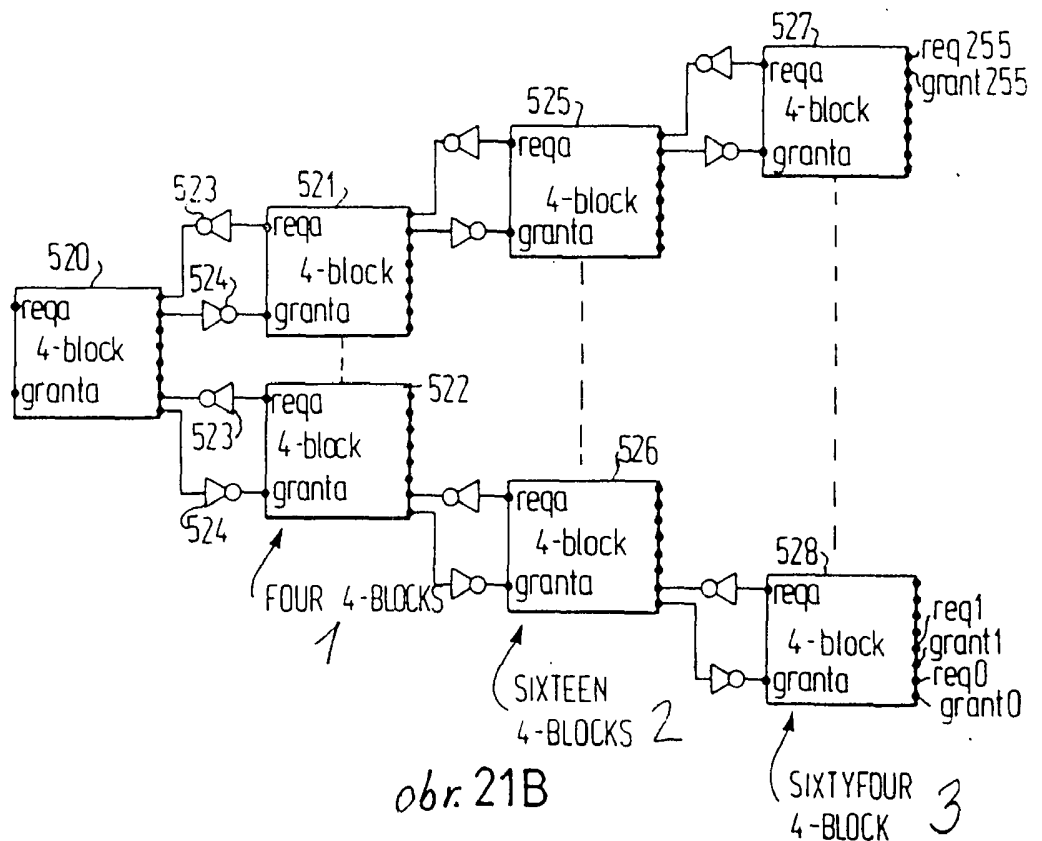
obr.19

16/37



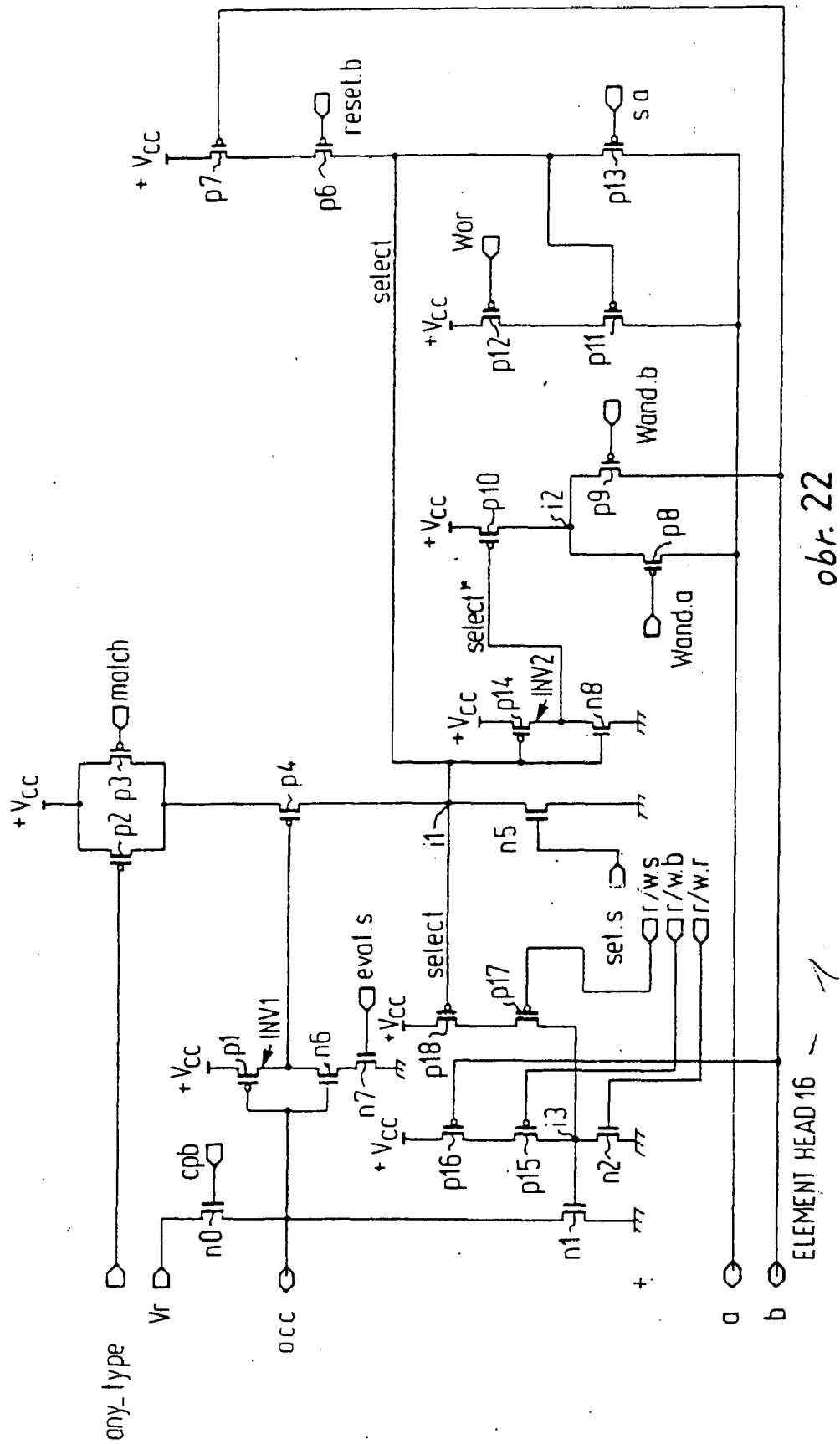


obr. 21A

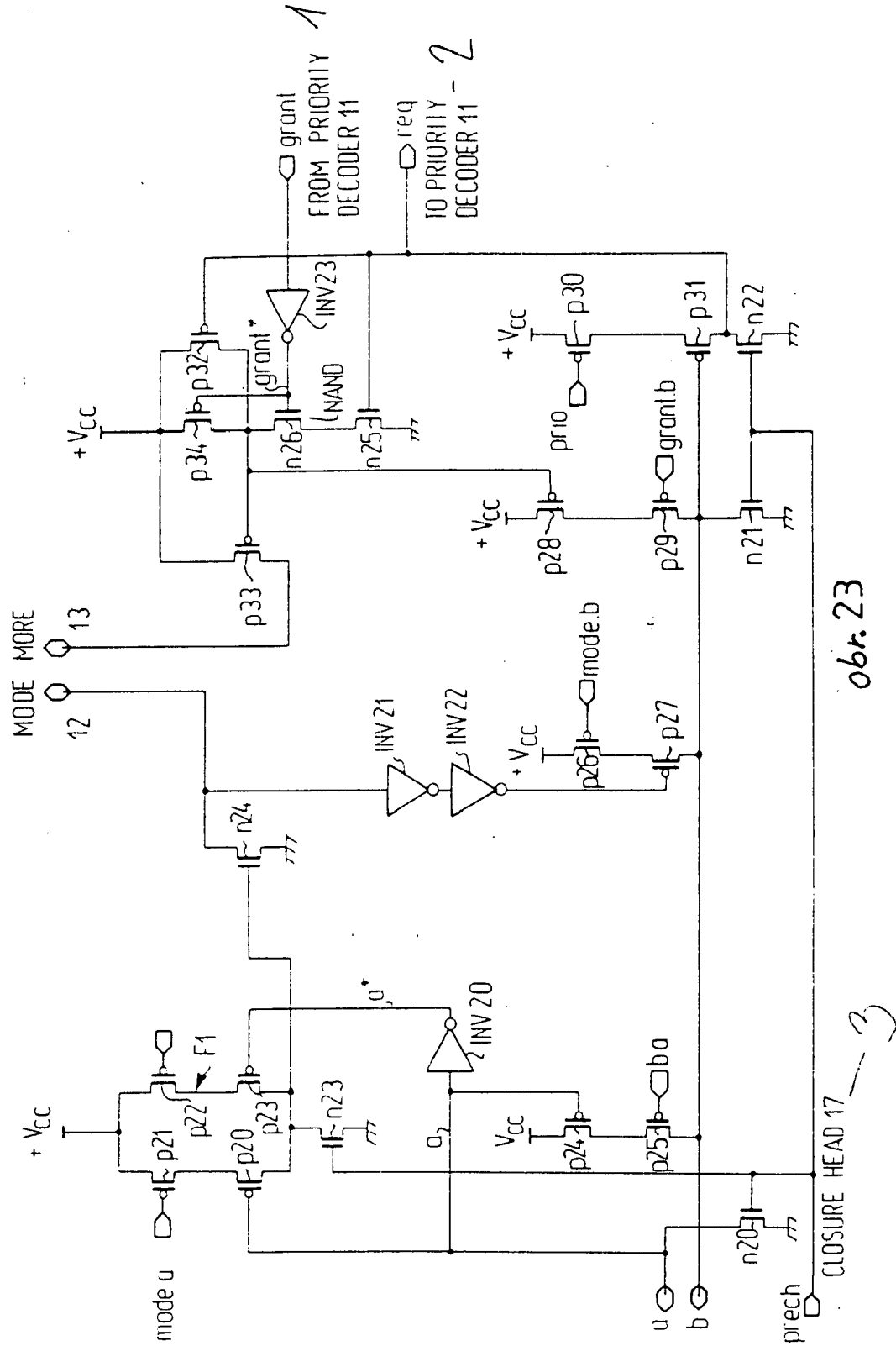


obr. 21B

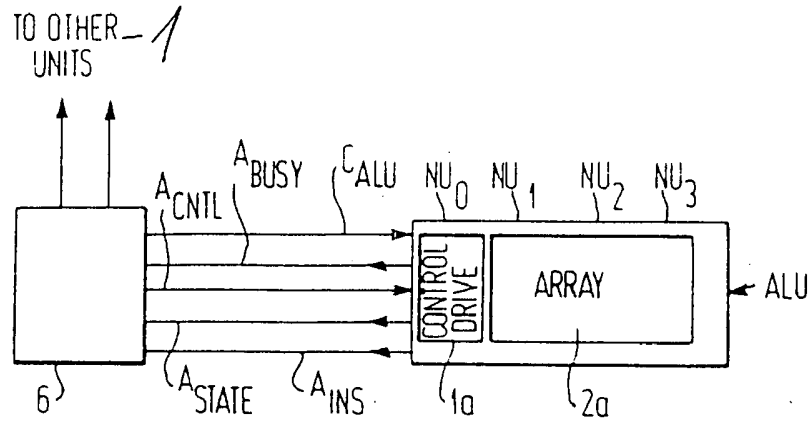
18/37



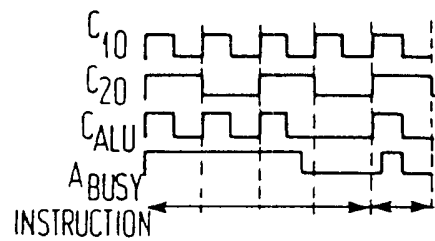
19/37



20/37



obr. 24A

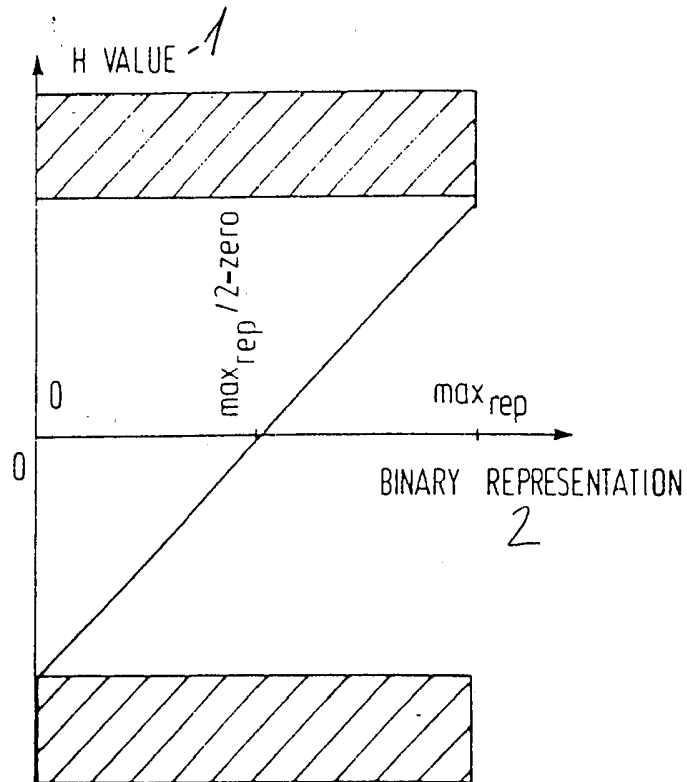


obr. 24B

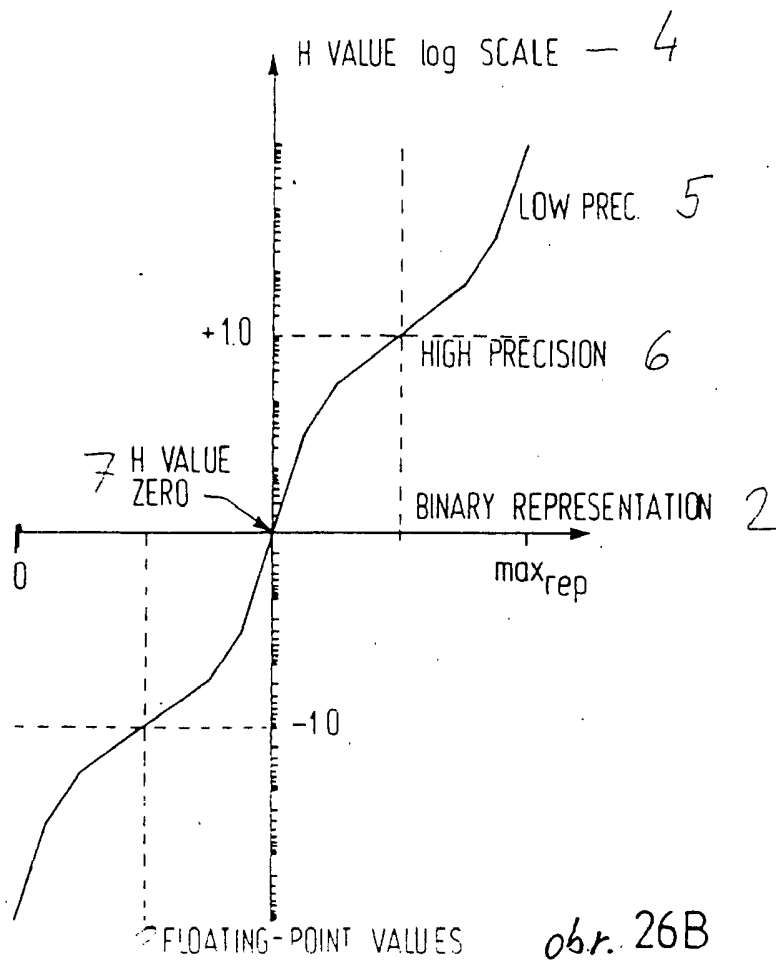


obr. 25

21/37



3 INTEGER VALUES obr. 26A



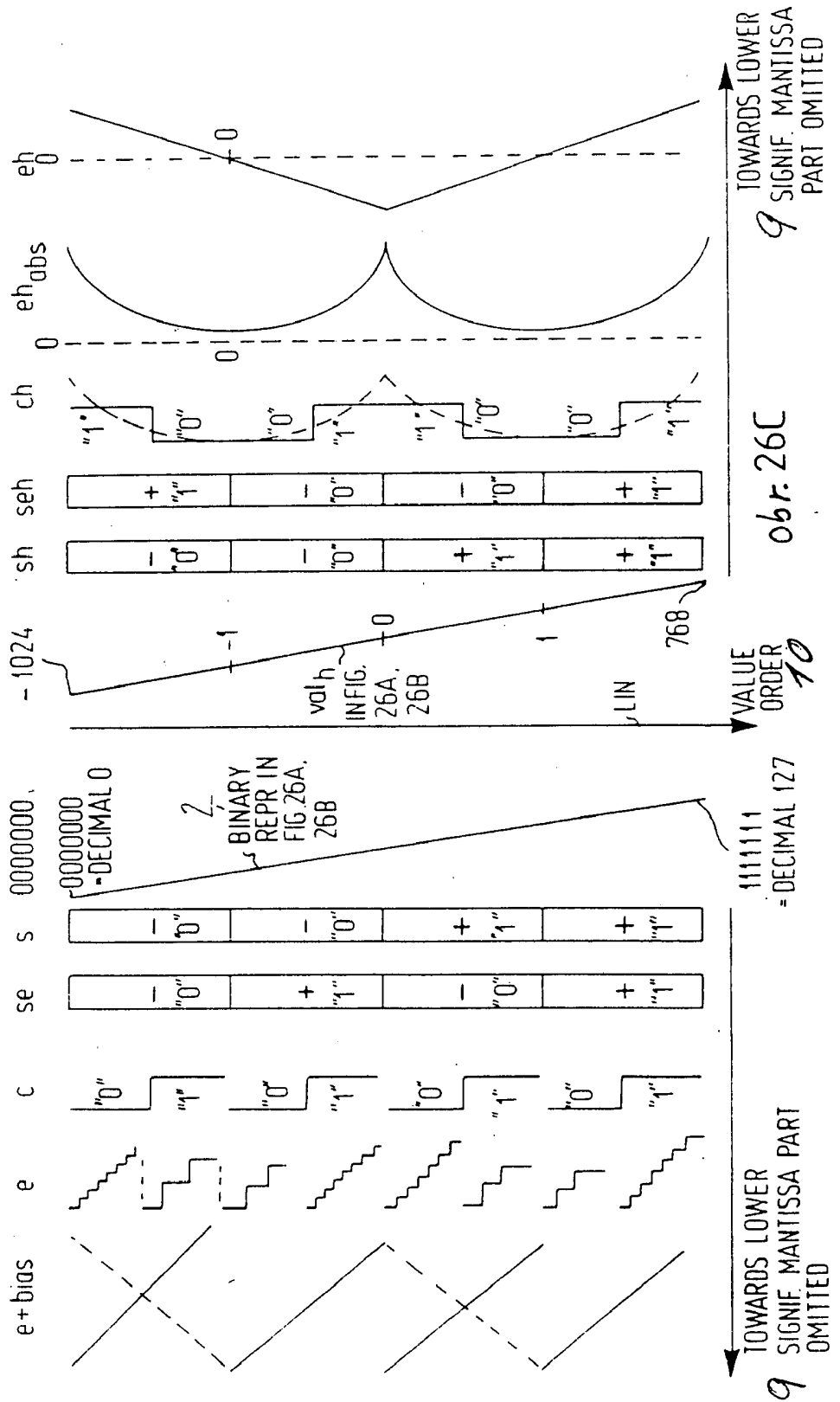
FLOATING-POINT VALUES obr. 26B

22/37

8 FLOATING-POINT REPRESENTATION

2 BINARY REPRESENTATION

1-H VALUES



2

ORDINARY		DECIMAL REPR. OF		BIN REPR.		s,sh,	se,seh,	c,	ch,	e,	e,		eh,	m,	md,	mh,	vdh,
BINARY REPR.	BIN REPR.	>	>	>	>	>	>	>	>	>	000,	000,	1001	0,000	0,000	-2	-1024
0000000	0	-	-	-	-	0	0	0	0	0	000,	000,	1001	0,000	0,000	-2	-1024
0000001	1	-	-	-	-	0	0	0	0	0	000,	000,	1001	1,000	1,000	-1.5	-768
0000010	2	-	-	-	-	0	0	0	0	1	001,	001,	1000	0,000	0,000	-1.5	-512
0000011	3	-	-	-	-	0	0	0	0	1	001,	001,	1000	1,000	1,000	-1.5	-384
0000100	4	-	-	-	-	0	0	0	0	1	010,	010,	0111	0,000	0,000	-1.5	-256
0000101	5	-	-	-	-	0	0	0	0	1	010,	010,	0111	1,000	1,000	-1.5	-192
0000110	6	-	-	-	-	0	0	0	0	1	011,	011,	0110	0,000	0,000	-1.5	-128
0000111	7	-	-	-	-	0	0	0	0	1	011,	011,	0110	1,000	1,000	-1.5	-96
0001000	8	-	-	-	-	0	0	0	0	1	100,	100,	0101	0,000	0,000	-1.5	-64
0001001	9	-	-	-	-	0	0	0	0	1	100,	100,	0101	1,000	1,000	-1.5	-48
0001010	10	-	-	-	-	0	0	0	0	1	101,	101,	0100	0,000	0,000	-1.5	-32
0001011	11	-	-	-	-	0	0	0	0	1	101,	101,	0100	1,000	1,000	-1.5	-24
0001100	12	-	-	-	-	0	0	0	0	1	110,	110,	0011	0,000	0,000	-1.5	-16
0001101	13	-	-	-	-	0	0	0	0	1	110,	110,	0011	1,000	1,000	-1.5	-12
0001110	14	-	-	-	-	0	0	0	0	1	111,	111,	0010	0,000	0,000	-1.5	-8
0001111	15	-	-	-	-	0	0	0	0	1	111,	111,	0010	1,000	1,000	-1.5	-6
0010000	16	-	-	-	-	0	0	1	0	0	000,	000,	0001	0,000	0,000	-2	-4
0010001	17	-	-	-	-	0	0	1	0	0	000,	000,	0001	1,000	1,000	-1.875	-3.75
0010010	18	-	-	-	-	0	0	1	0	0	000,	000,	0001	2,000	2,000	-1.75	-3.5
0010011	19	-	-	-	-	0	0	1	0	0	000,	000,	0001	3,000	3,000	-1.625	-3.25
0010100	20	-	-	-	-	0	0	1	0	0	000,	000,	0001	4,000	4,000	-1.5	-3
0010101	21	-	-	-	-	0	0	1	0	0	000,	000,	0001	5,000	5,000	-1.375	-2.75
0010110	22	-	-	-	-	0	0	1	0	0	000,	000,	0001	6,000	6,000	-1.25	-2.5
0010111	23	-	-	-	-	0	0	1	0	0	000,	000,	0001	7,000	7,000	-1.125	-2.25
0011000	24	-	-	-	-	0	0	1	0	1	001,	001,	0000	0,000	0,000	-2	-2
0011001	25	-	-	-	-	0	0	1	0	1	001,	001,	0000	1,000	1,000	-1.875	-1.875
0011010	26	-	-	-	-	0	0	1	0	1	001,	001,	0000	2,000	2,000	-1.75	-1.75
0011011	27	-	-	-	-	0	0	1	0	1	001,	001,	0000	3,000	3,000	-1.625	-1.625
0011100	28	-	-	-	-	0	0	1	0	1	001,	001,	0000	4,000	4,000	-1.5	-1.5
0011101	29	-	-	-	-	0	0	1	0	1	001,	001,	0000	5,000	5,000	-1.375	-1.375
0011110	30	-	-	-	-	0	0	1	0	1	001,	001,	0000	6,000	6,000	-1.25	-1.25
0011111	31	-	-	-	-	0	0	1	0	1	001,	001,	0000	7,000	7,000	-1.125	-1.125

obr 27A

24/37

1 2

ORDINARY DECIMAL REPR. OF

BINARY REPR. BIN. REPR.

ORDINARY BINARY REPR.	DECIMAL REPR. OF BIN. REPR.	s,sh	se,seh	c	ch	e	eh	e	eh	m _d	m _h	vd _h
0100000	32	-	1	0	0	0	-1	000,0000	-1	0,000	-2	-1
0100001	33	-	1	0	0	0	-1	000,0000	-1	1,001	-1.875	-.9375
0100010	34	-	1	0	0	0	-1	000,0000	-1	2,010	-1.775	-.875
0100011	35	-	1	0	0	0	-1	000,0000	-1	3,011	-1.625	-.8125
0100100	36	-	1	0	0	0	-1	000,0000	-1	4,100	-1.5	-.75
0100101	37	-	1	0	0	0	-1	000,0000	-1	5,101	-1.375	-.6875
0100110	38	-	1	0	0	0	-1	000,0000	-1	6,110	-1.25	-.625
0100111	39	-	1	0	0	0	-1	000,0000	-1	7,111	-1.125	-.5625
0101000	40	-	1	0	0	1	-2	001,0000	-2	0,000	-2	-.5
0101001	41	-	1	0	0	1	-2	001,0000	-2	1,001	-1.875	-.46875
0101010	42	-	1	0	0	1	-2	001,0000	-2	2,010	-1.775	-.4375
0101011	43	-	1	0	0	1	-2	001,0000	-2	3,011	-1.625	-.40625
0101100	44	-	1	0	0	1	-2	001,0000	-2	4,100	-1.5	-.375
0101101	45	-	1	0	0	1	-2	001,0000	-2	5,101	-1.375	-.34375
0101110	46	-	1	0	0	1	-2	001,0000	-2	6,110	-1.25	-.3125
0101111	47	-	1	0	0	1	-2	001,0000	-2	7,111	-1.125	-.28125
0110000	48	-	1	1	1	0	-3	000,0000	-3	0,000	-2	-.25
0110001	49	-	1	1	1	0	-3	000,0000	-3	1,001	-1.5	-.1875
0110010	50	-	1	1	1	1	-4	001,0000	-4	0,000	-2	-.125
0110011	51	-	1	1	1	1	-4	001,0000	-4	1,001	-1.5	-.09375
0110100	52	-	1	1	1	2	-5	010,0000	-5	0,000	-2	-.0625
0110101	53	-	1	1	1	2	-5	010,0000	-5	1,001	-1.5	-.046875
0110110	54	-	1	1	1	3	-6	011,0000	-6	0,000	-2	-.03125
0110111	55	-	1	1	1	3	-6	011,0000	-6	1,001	-1.5	-.0234375
0111000	56	-	1	1	1	4	-7	100,0000	-7	0,000	-2	-.015625
0111001	57	-	1	1	1	4	-7	100,0000	-7	1,001	-1.5	-.0117188
0111010	58	-	1	1	1	5	-8	101,0000	-8	0,000	-2	-.0078125
0111011	59	-	1	1	1	5	-8	101,0000	-8	1,001	-1.5	-.00585938
0111100	60	-	1	1	1	6	-9	110,0000	-9	0,000	-2	-.00390625
0111101	61	-	1	1	1	6	-9	110,0000	-9	1,001	-1.5	-.00292969
0111110	62	-	1	1	1	7	-10	111,0000	-10	0,000	-2	-.00195312
0111111	63	-	1	1	1	7	-10	111,0000	-10	1,001	-1.5	-.00146484

obr. 27B

25/37

ORDINARY BINARY REPR.	DECIMAL REPR OF BIN. REPR.	s,sh>	se>	seh>	c>	ch>	e>	eh>	e>	eh>	m _d >	m _h >	valh>
1000000	64	1	0	0	0	1	0	-10	000	0000	0,000	1	0
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	1.125	.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	1.125	.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

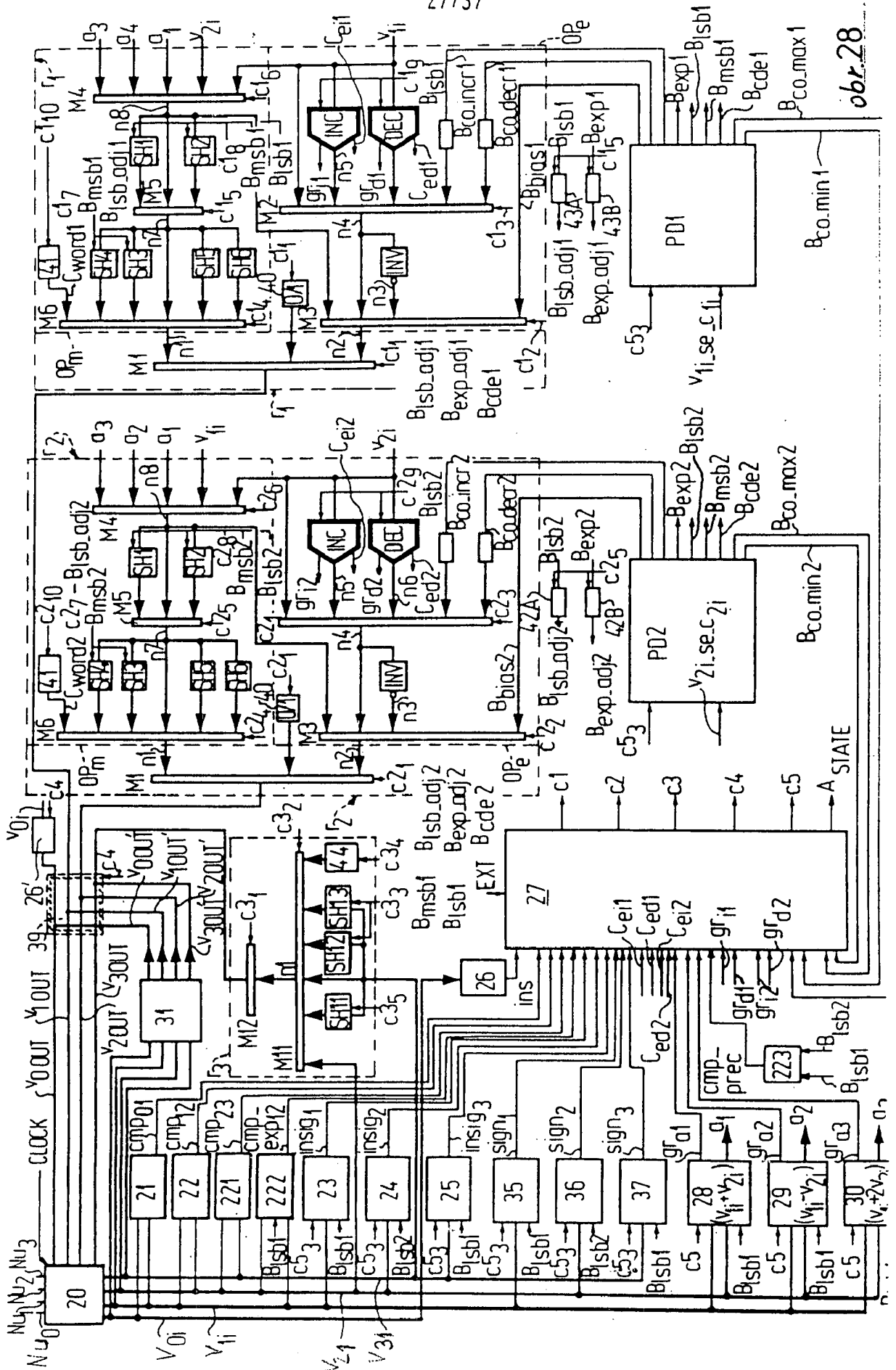
abr. 27C

26/37

1	2	ORDINARY BINARY REPR.	DECIMAL REPR. OF BIN. REPR.	s,sh>	se>	seh>	c>	ch>	e>	eh>	e>	eh>	m>	md>	mh>	valh>
11000000	96	->	->	1	1	1	0	0	0	0	000	0000	0	0,000	1	1
11000001	97	->	->	1	1	1	0	0	0	0	000	0000	1	0,001	1.125	1.125
11000010	98	->	->	1	1	1	0	0	0	0	000	0000	2	0,010	1.25	1.25
11000011	99	->	->	1	1	1	0	0	0	0	000	0000	3	0,011	1.375	1.375
11000100	100	->	->	1	1	1	0	0	0	0	000	0000	4	0,100	1.5	1.5
11000101	101	->	->	1	1	1	0	0	0	0	000	0000	5	0,101	1.625	1.625
11000110	102	->	->	1	1	1	0	0	0	0	000	0000	6	0,111	1.75	1.75
11000111	103	->	->	1	1	1	0	0	0	0	000	0000	7	0,111	1.875	1.875
11010000	104	->	->	1	1	1	0	0	0	1	001	0001	0	0,000	2	2
11010001	105	->	->	1	1	1	0	0	0	1	001	0001	1	0,001	2.25	2.25
11010010	106	->	->	1	1	1	0	0	0	1	001	0001	2	0,010	2.5	2.5
11010011	107	->	->	1	1	1	0	0	0	1	001	0001	3	0,011	2.75	2.75
11011000	108	->	->	1	1	1	0	0	0	1	001	0001	4	0,100	3	3
11011001	109	->	->	1	1	1	0	0	0	1	001	0001	5	0,101	3.25	3.25
11011010	110	->	->	1	1	1	0	0	0	1	001	0001	6	0,110	3.5	3.5
11011011	111	->	->	1	1	1	0	0	0	1	001	0001	7	0,111	3.75	3.75
11100000	112	->	->	1	1	1	1	0	0	0	000	0010	0	0,000	4	4
11100001	113	->	->	1	1	1	1	0	0	0	000	0010	1	0,001	5	5
11100010	114	->	->	1	1	1	1	0	0	0	001	0011	0	0,000	6	6
11100011	115	->	->	1	1	1	1	0	0	0	001	0011	1	0,001	8	8
11101000	116	->	->	1	1	1	1	0	0	1	010	0100	0	0,000	12	12
11101001	117	->	->	1	1	1	1	0	0	1	010	0100	1	0,001	16	16
11101010	118	->	->	1	1	1	1	0	0	1	010	0100	2	0,001	24	24
11101011	119	->	->	1	1	1	1	0	0	1	011	0101	0	0,000	32	32
11110000	120	->	->	1	1	1	1	0	0	1	011	0101	1	0,001	48	48
11110001	121	->	->	1	1	1	1	0	0	1	100	0110	0	0,000	64	64
11110010	122	->	->	1	1	1	1	0	0	1	100	0110	1	0,001	96	96
11110011	123	->	->	1	1	1	1	0	0	1	101	0111	0	0,000	128	128
11111000	124	->	->	1	1	1	1	0	0	1	101	0111	1	0,001	192	192
11111001	125	->	->	1	1	1	1	0	0	1	110	1000	0	0,000	256	256
11111010	126	->	->	1	1	1	1	0	0	1	110	1000	1	0,001	384	384
11111011	127	->	->	1	1	1	1	0	0	1	111	1001	0	0,000	512	512
		->	->								111	1001	1	0,001	768	768

obr 27D

27/37

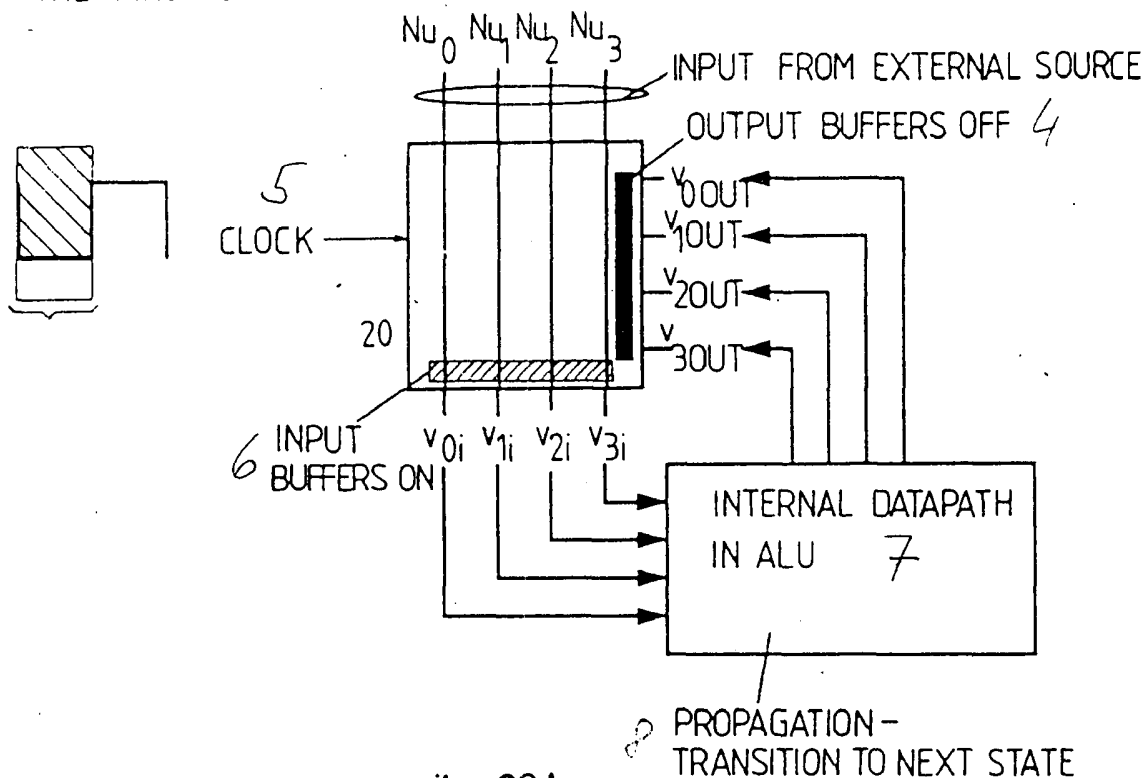


obr.28

28/37

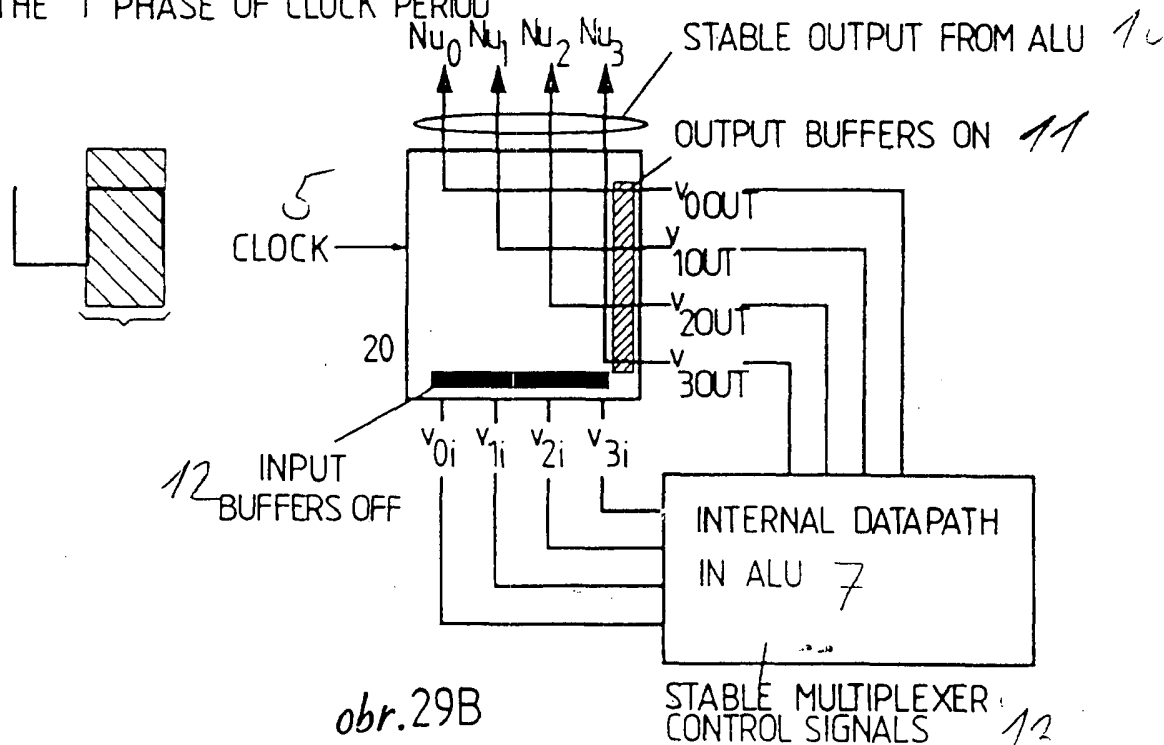
1 CLOCKING OF INPUT/OUTPUT BUFFER 20

2 THE FIRST 0 PHASE OF CLOCK PERIOD OF AN INSTRUCTION



obr.29A

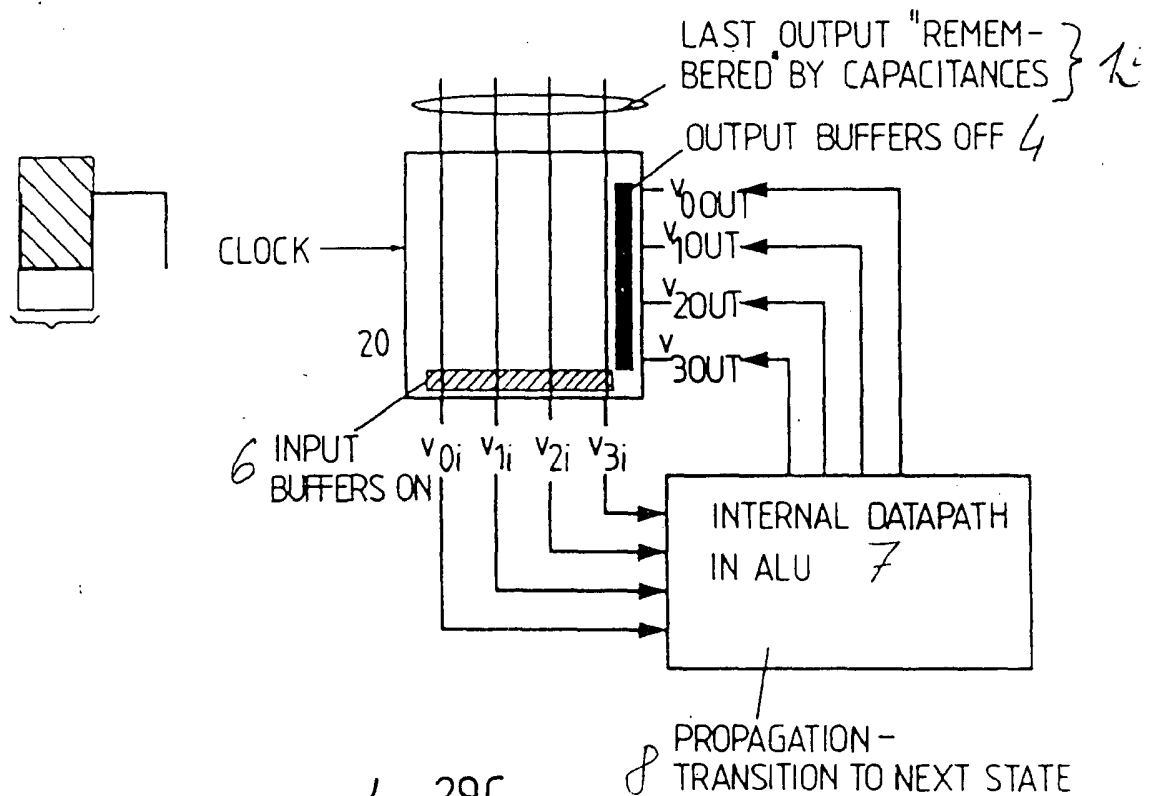
THE 1 PHASE OF CLOCK PERIOD



obr.29B

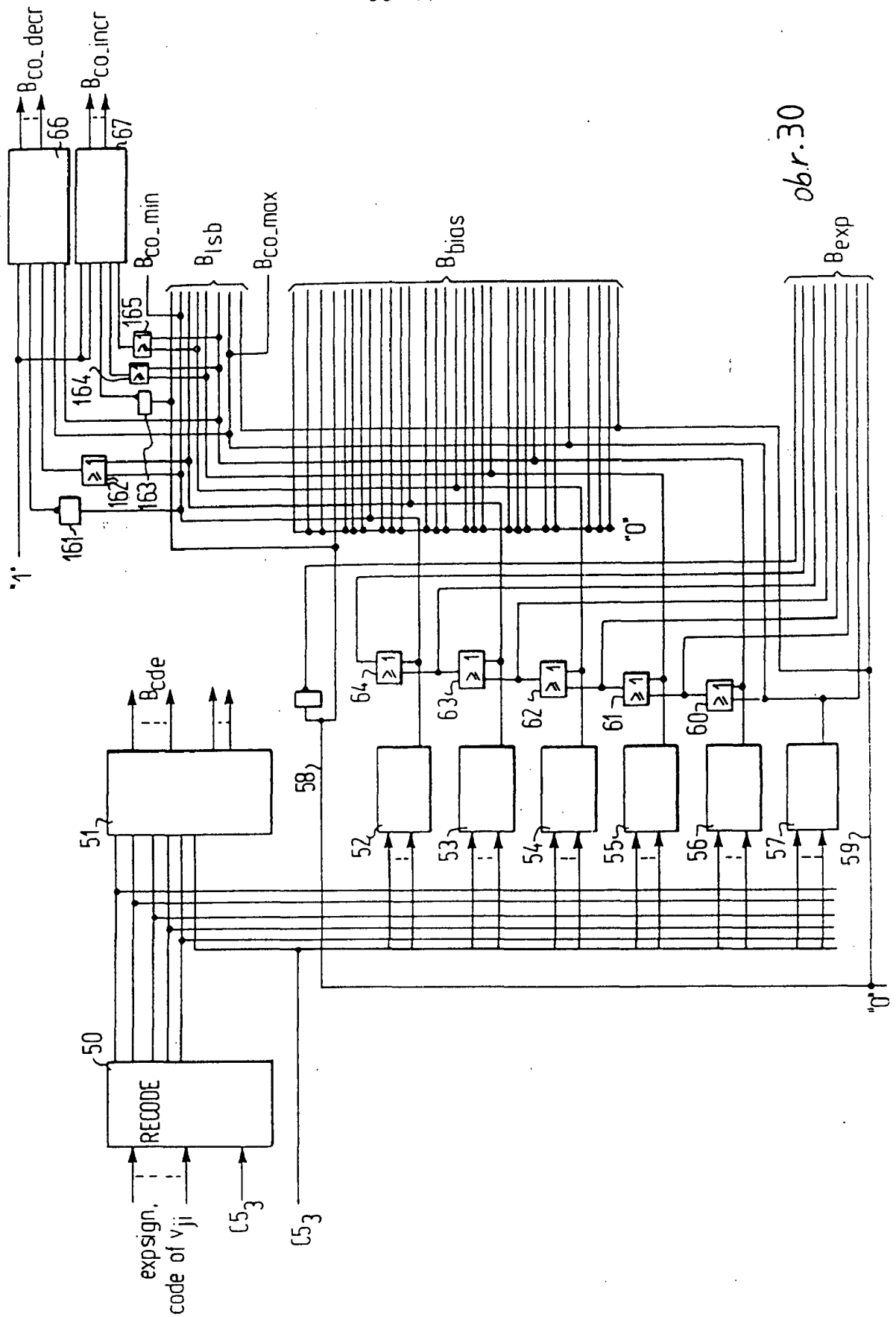
29/37

14 THE REST OF 0 PHASES OF CLOCK PERIODS OF AN MULTIPLE
CYCLE INSTRUCTION



obr. 29C

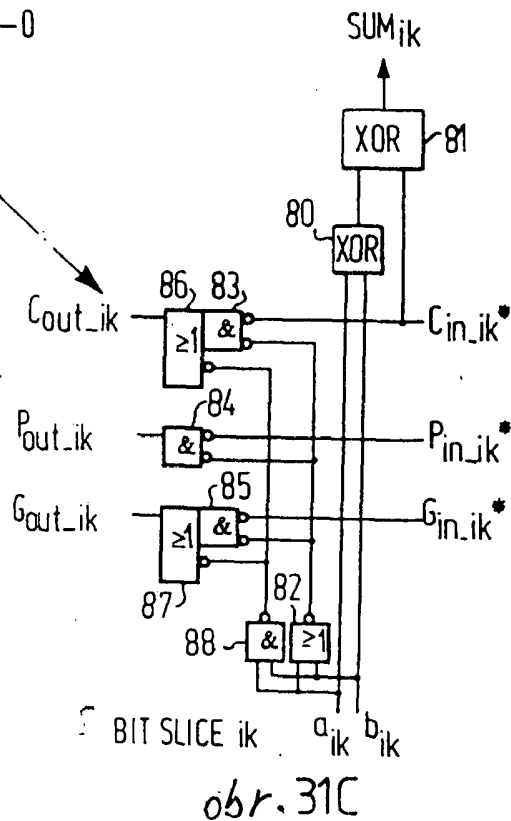
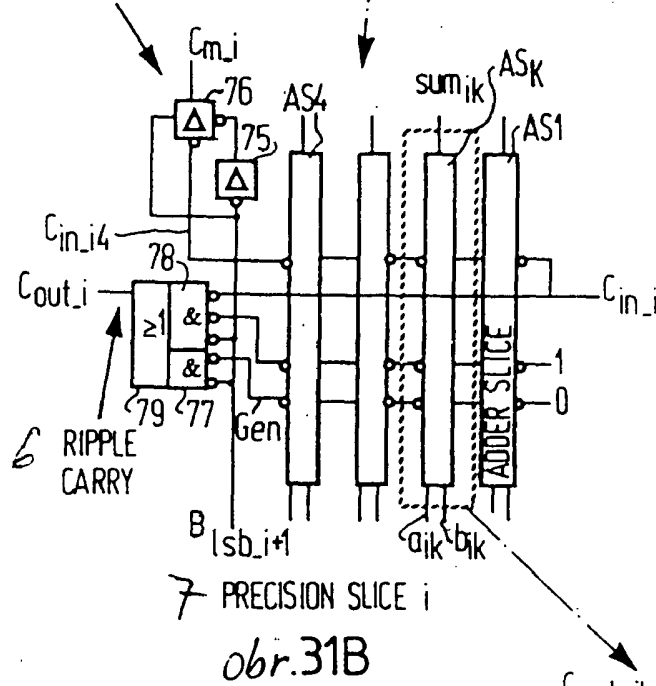
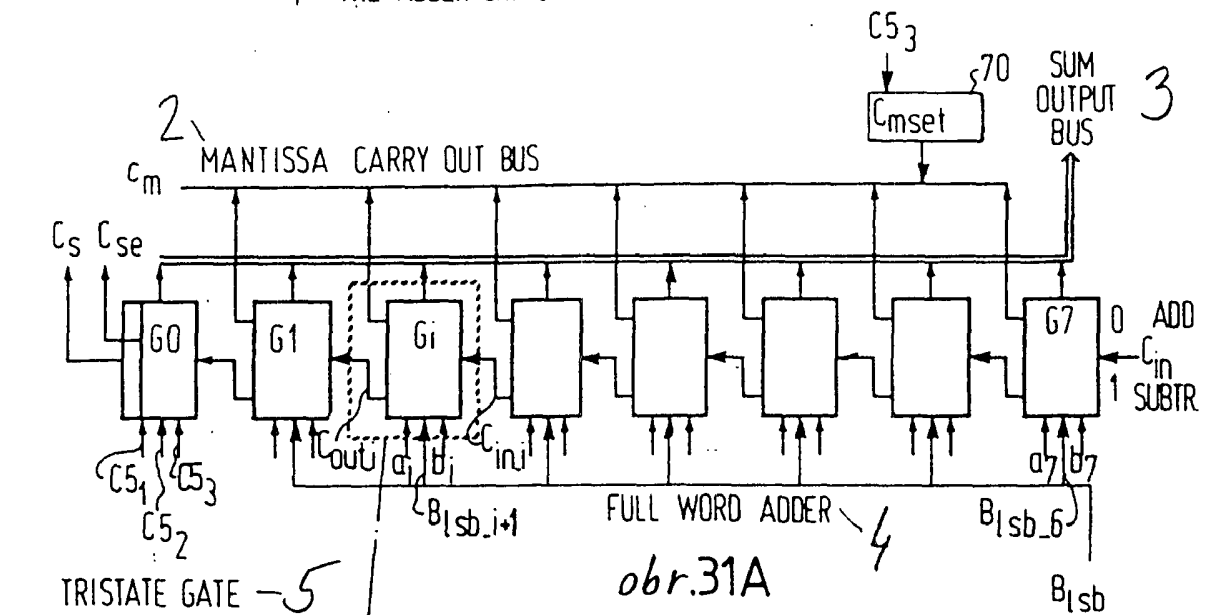
30 / 37



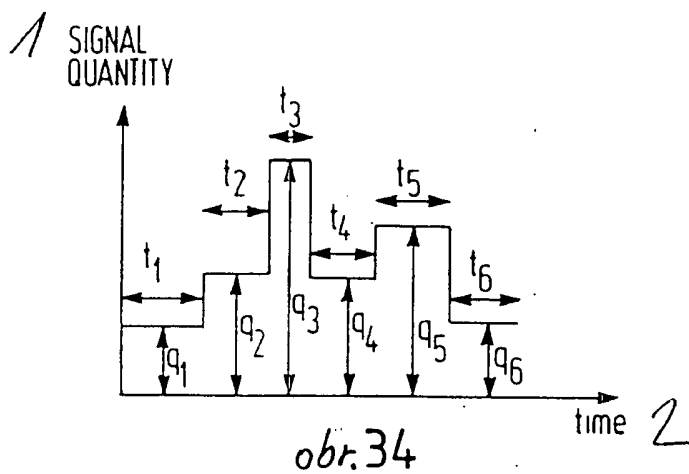
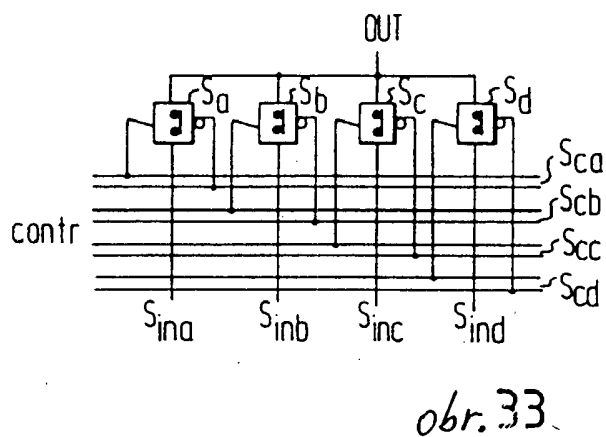
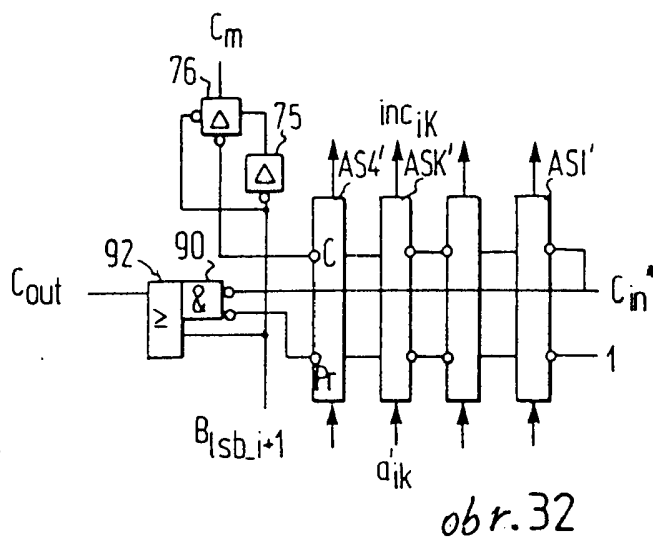
NO 92/02070

31/37

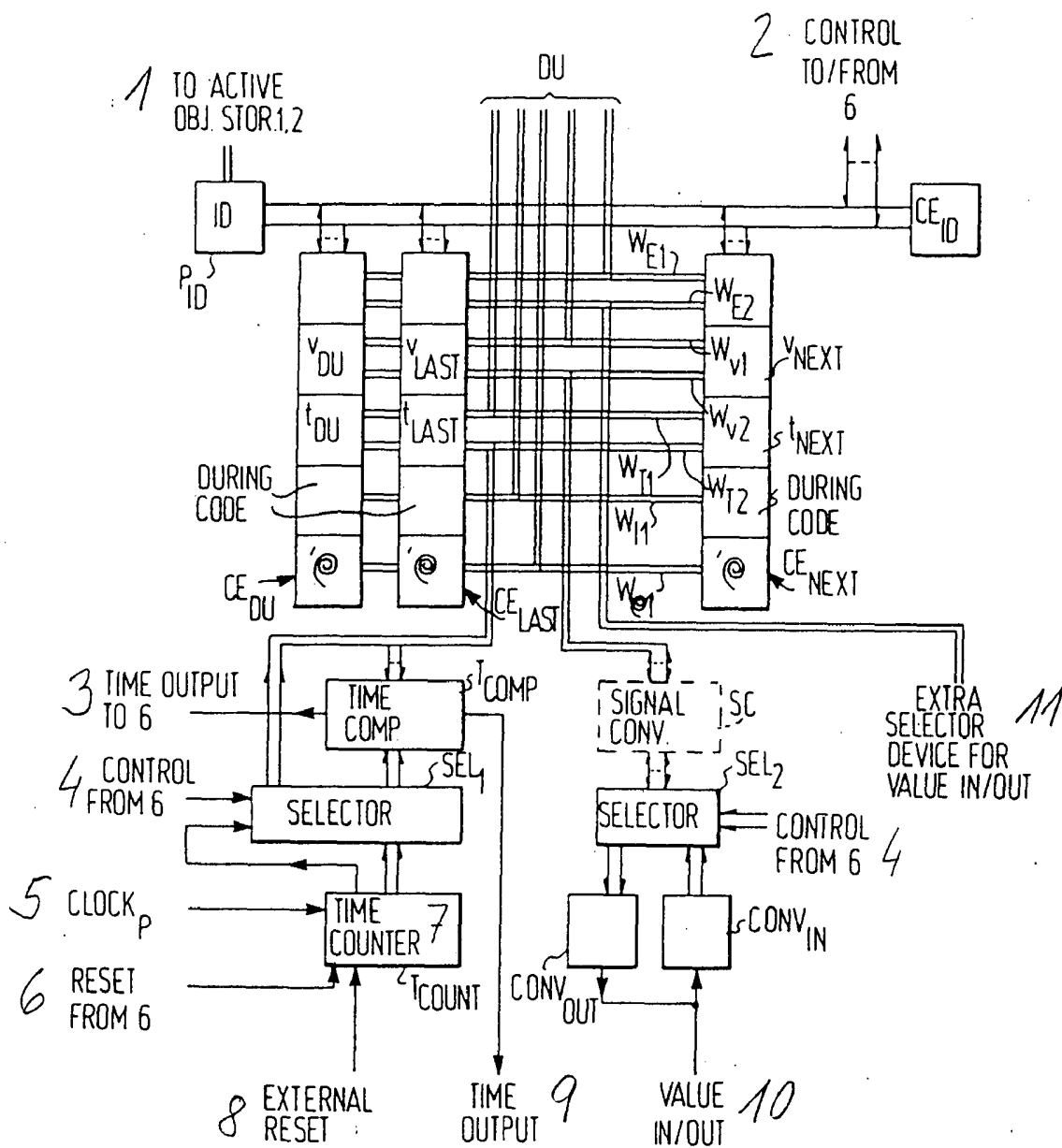
1 - THE ADDER GRP UNIT



32/37

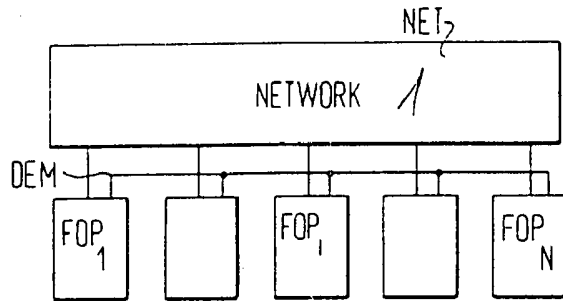


33/37

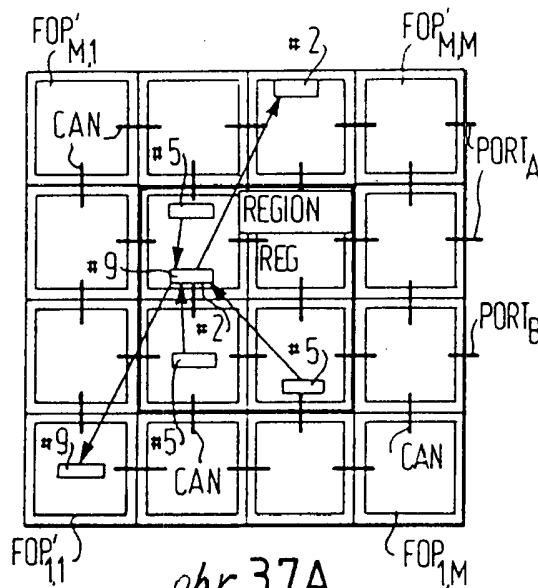


obr.35

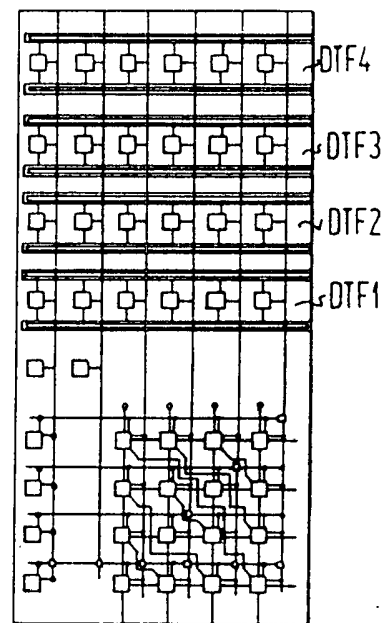
34/37



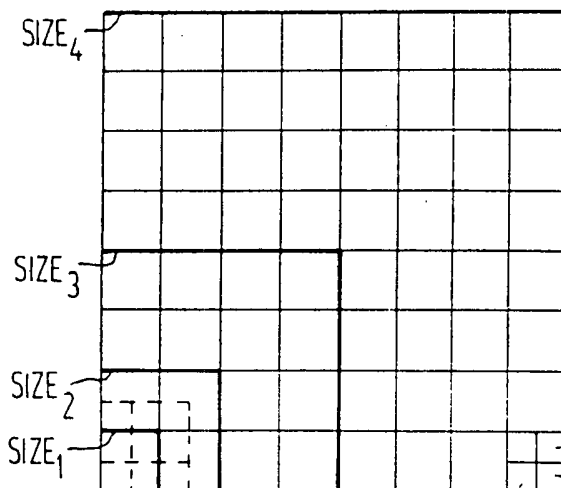
obr.36



obr.37A



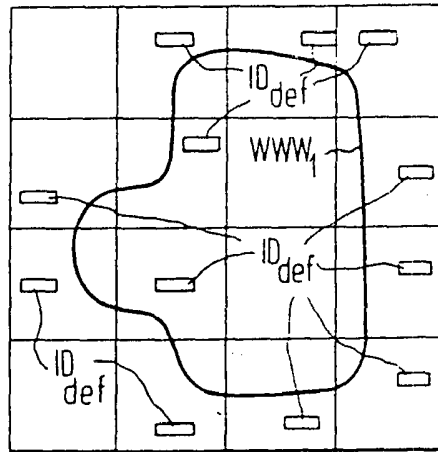
obr.37B



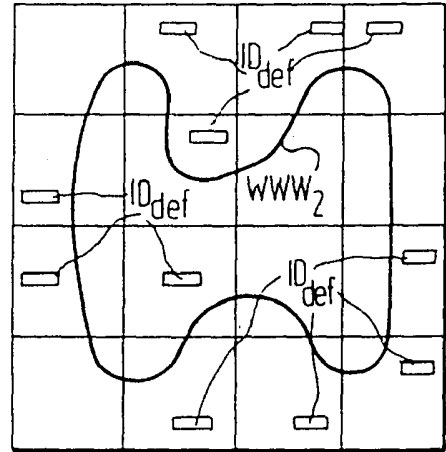
obr.37C

APPROPRIATE DIVISIONS INTO REGIONS
OF A SQUARE FIELD SECOND ORDER
PROCESSOR INCLUDING 256 FOPs: } 1
256 SIZE 0 REGIONS
64 SIZE 1 REGIONS } 2
16 SIZE 2 REGIONS
4 SIZE 3 REGIONS
1 SIZE 4 REGION

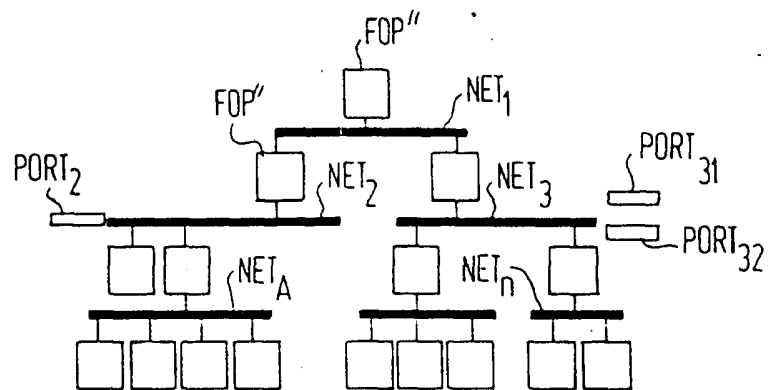
35/37



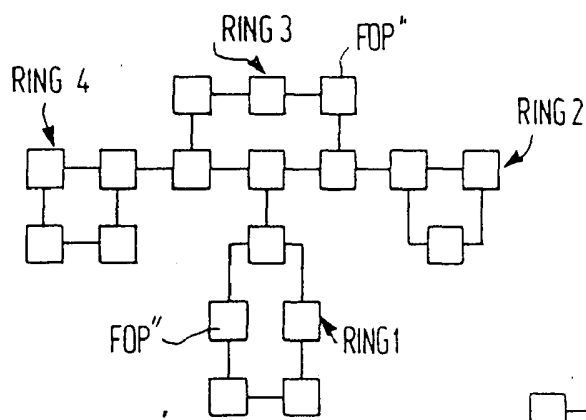
obr.37D



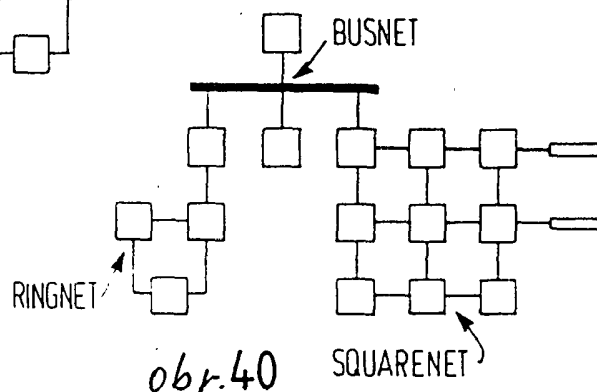
obr.37E



obr.38

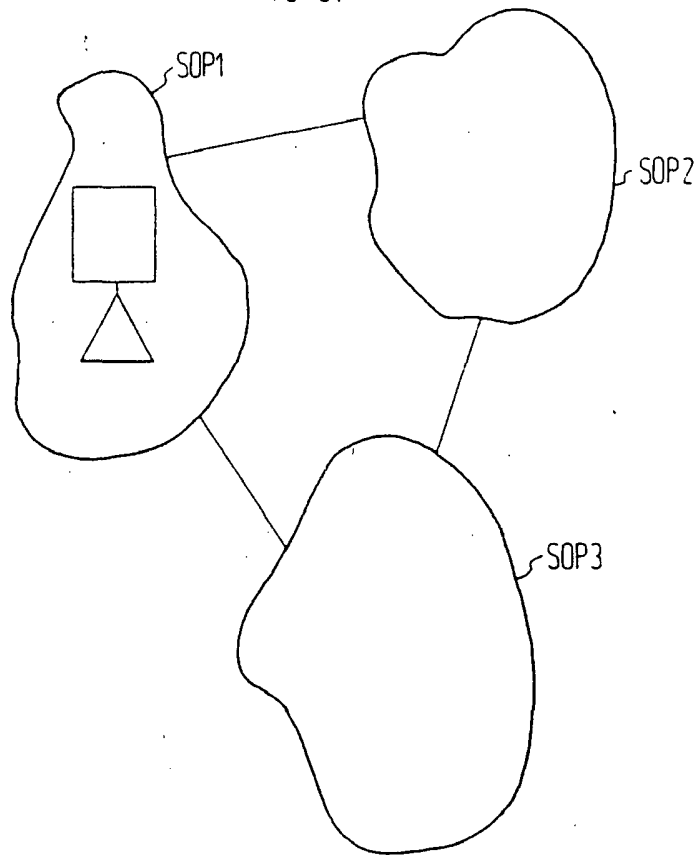


obr.39



obr.40

36/37



obr. 41

121	-----	623	-----
-----	-------	-----	-------

id_{gf1}

id_{gs}

999	-----	623	-----
-----	-------	-----	-------

id_{gf2}

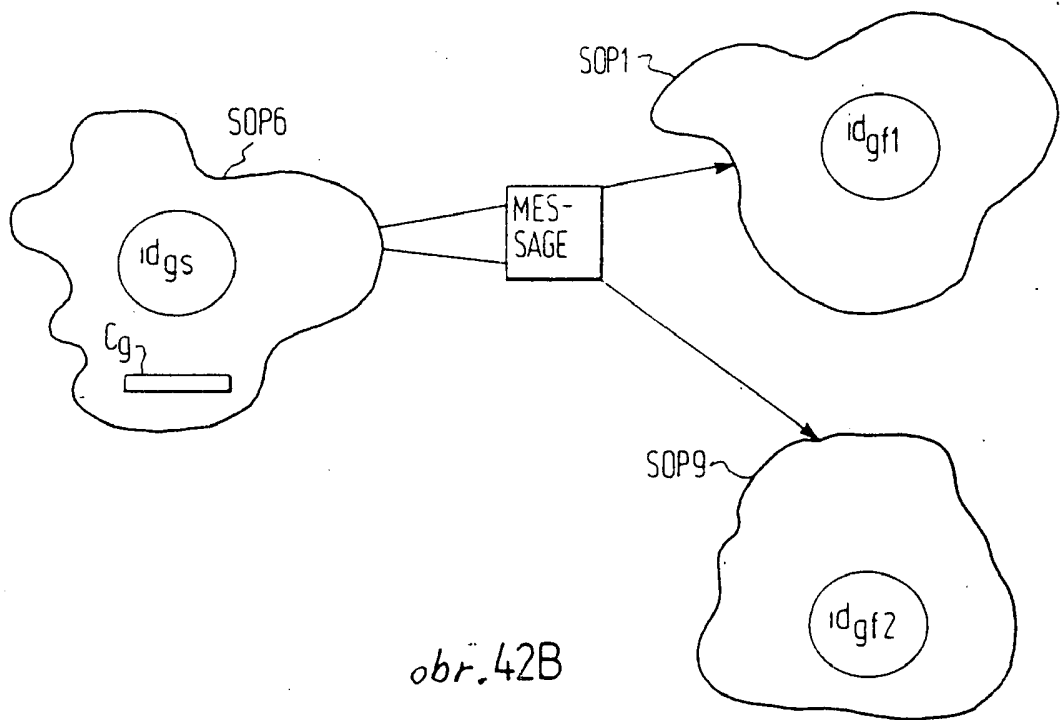
id_{gs}

623	-----
-----	-------

id_{gs}

obr. 42A

37/37



obr.42B



obr.42C