PROCESOR S PODATKOVNO PRETOKOVNO ARHITEKTURO

Institut Jožef Stefan, Ljubljana
UDK: 681.3.02

Jurij Šilc, Borut Robič

Članek predstavlja prvi procesor s podatkovno pretokovno arhitekturo. Notranja krožna pipeline organizacija, ki temelji na podatkovnem vodenju, daje procesorju veliko moč. Bogat nabor ukazov je prirejen tako, da je procesor zelo uporaben za hitro obdelavo slikovnih in govornih signalov. Potreba po takšnih obdelavah se pojavlja v računalnikih pete generacije.

Data Flow Architecture Based Processor - A new data flow architecture based processor is described. The processor employs token based data flow and pipelined architecture to achieve a very high throughput rate in the realm of digital image and speech processing.

1. UVOD

Pri načrtovanju sistemov za obdelavo slik smo običajno prisiljeni poiskati kompromis med hitrostjo in fleksibilnostjo sistema. Okornost in počasnost sistema, ki ga sestavljata miniračunalnik za obdelavo slike in masovni pomnilnik za njeno shranjevanje, sta nesprejemljivi za delo v realnem času. Z dodatkom posebne materialne opreme se hitrost obdelave poveča, toda žal na račun fleksibilnosti, saj vsaka sprememba programske opreme narekuje spremembo materialne opreme. Omenjeni razkorak med hitrostjo in fleksibilnostjo sistema je moč omiliti z uporabo programabilnega slikovnega procesorja, ki se odlikuje s podatkovno vodeno arhitekturo.

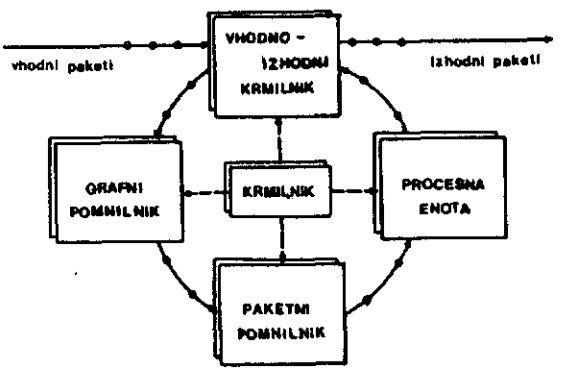
2. PODATKOVNO PRETOKOVNI PROCESOR

V nasprotju s von Neumannovimi procesorji, ki delujejo na podlagi dostave ukazov, temelji delo podatkovno pretokovnega procesorja na zbiranju in obdelavi paketov (tokens).

2.1. Osnovna arhitektura procesorja

Podatkovno pretokovni procesor ne rabi dostave ukaza. Namesto tega vsebuje grafni pomnilnik (tabeli povezav in točk), v katerega se pred pričetkom izvrševanja vpiše programski podgraf. Pretok podatkov se vrši s pomočjo paketov, ki vsebujejo naslov procesorja, identifikator, podatkovno ter krmilno polje. Med pretokom po procesorju paket še nekajkrat spremeni vsebino in dolžino, kot bo razvidno iz kasnejšega opisa.

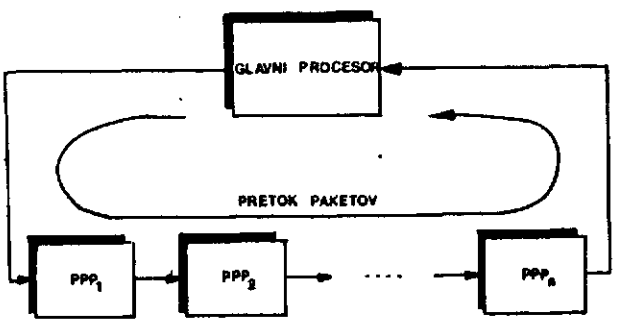
Notranja krožna pipeline arhitektura (Slika 1) omogoča procesni enoti neprekinjeno delovanje. Procesna enota vsebuje množilnik in ALU, ki omogoča standardne aritmetično logične operacije. Nabor ukazov je širši kot pri večini klasičnih procesorjev.



Slika 1: Krožna pipeline arhitektura podatkovno pretokovnega procesorja.

2.2. Večprocesorski podatkovno pretokovni sistem

Večprocesorski podatkovno pretokovni sistem sestavlja kaskada podatkovno pretokovnih procesorjev, povezanih z glavnim procesorjem, kot je prikazano na Sliki 2.

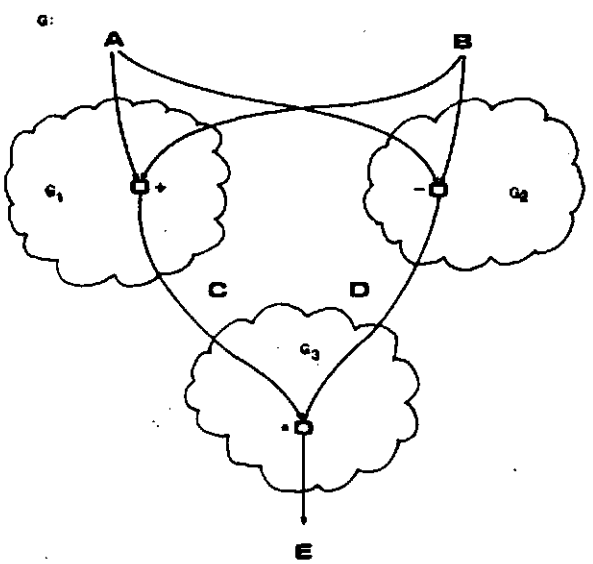


Slika 2: Podatkovno pretokovni računalnik.

Komunikacija med podatkovno pretokovnim procesorjem in okolico poteka s pomočjo vhodno-izhodnega krmilnika. Glavni procesor pošlje paket podatkovno pretokovnim procesorjem. Iz naslovnega polja paketa podatkovno pretokovni procesor ugotovi, če je paket namenjen njemu ter ga v tem primeru sprejme, izloči naslovno polje in pošlje v pretok - najprej v grafni pomnilnik (tabelo povezav). Paket, ki ni namenjen danemu procesorju, se nespremenjen pošlje v istem ciklu preko vhodno-izhodnega krmilnika naslednjemu procesorju. Procesor je tako za tuje pakete transparenten. Torej vsak podatkovno pretokovni procesor zbira svoje pakete.

2.3. Delovanje podatkovno pretokovnega sistema

Delovanje podatkovno pretovnega računalnika, kot ga prikazuje Slika 2, ilustrirajmo z naslednjim primerom. Dan naj bo program za izračun E = (A+B)*(A-B), opisan s podatkovno pretokovnim programskim grafom G (Slika 3). Graf G lahko razbijemo na tri podgrafe G_1, G_2 in G_3. Ker se lahko izvršujeta G_1 in G_2 vzporedno, ju je smiselno vpisati v dva različna podatkovno pretokovna procesorja P_1 in P_2. G_3 lahko vpišemo v katerikoli procesor, npr. v P_1. Program se prične izvrševati takoj, ko glavni procesor pošlje vhodna paketa A in B. Tako mora za izvršitev operacije C = A + B procesor P_1 prejeti paketa, ki nosita vrednosti A in B, šele nato lahko izvrši operacijo '+'. Vhodna paketa lahko prispeta v poljubnem zaporedju, saj je procesor sposoben razpoznavati pakete in jih hraniti v paketnem pomnilniku. Iz opisa podgrafa G_1, ki se nahaja v njegovem grafnem pomnilniku, P_1 ugotovi, da paketa A in B pripadata preko operacije '+' paketu C, zato ju združi in pošlje v procesno enoto. Vrednost, ki je rezultat izvršitve v procesni enoti, se vstavi v paket in opremi z oznako C. Istočasno se v procesorju P_2 izračuna paket D, ki ga P_2 pošlje glavnemu procesorju. Ker je C vhodni paket nove operacije, ki se mora izvršiti v istem procesorju P_1, se shrani v njegov paketni pomnilnik, dokler iz glavnega procesorja ne prispe paket D. Tedaj sta v procesorju P_1 oba paketa za izvršitev operacije '*' in že prej opisani postopek se ponovi. Vidimo, da zagotavlja pravilna izbira predhodno vpisanega podgrafa izkoriščanje vsebovane vzporednosti ter neprekinjeno delo procesorjev.



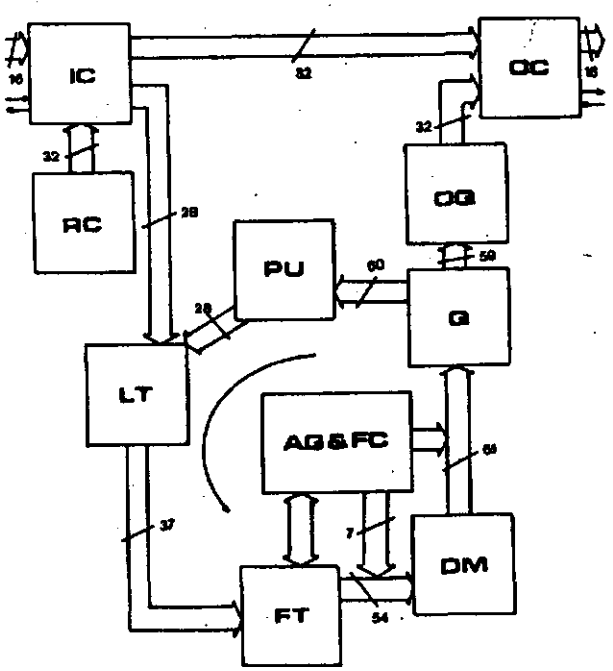
Slika 3: Podatkovno pretokovni graf za izračun E = (A+B)*(A-B).

3. PODATKOVNO PRETOKOVNI PROCESOR μPD7281

Primer podatkovno pretokovnega procesorja je NEC μPD7281, katerega moč temelji na krožno organizirani pipeline arhitekturi ter bogatem naboru ukazov. μPD7281 je prvi VLSI čip, ki deluje po načelih podatkovno pretokovne arhitekture [3]. Ta omogoča večjo učinkovitost procesorja v mnogih večprocesorskih aplikacijah, kot sta procesiranje slik ter razpoznavanje vzorcev na področju umetne inteligence, kjer se uporabljajo algoritmi za dvodomenzionalno konvolucijo, povečavo, pomanjšavo in rotacijo. Njegova učinkovitost postane očitna predvsem pri procesiranju slik v realnem času, kjer dodobra izkorišča vsebovane vzporednosti uporabljenih algoritmov. μPD7281 ni uporaben le pri procesiranju slik, temveč tudi pri zahtevnih numeričnih izračunih, kot so matrično matrično množenje, matrično vektorsko množenje, aritmetika s plavajočo vejico ter izračuni transcendentnih funkcij v realnem času.

3.1. Pipeline organizacija procesorja

Kot je prikazano na Sliki 4 sestavlja procesor deset funkcionalnih enot: vhodni krmilnik (IC), izhodni krmilnik (OC), tabela povezav (LT), tabela točk (FT), paketni pomnilnik (DM), vrsta (Q), procesna enota (PU), izhodna vrsta (OQ), generator naslovov in krmilnik pretoka (AG&FC) ter osveževalni krmilnik (RC).

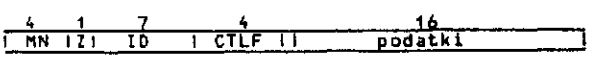


IC: vhodni krmilnik (Input Controller)
OC: izhodni krmilnik (Output Controller)
LT: tabela povezav (Link Table)
FT: tabela točk (Function Table)
DM: paketni pomnilnik (Data Memory)
Q: vrsta (Queue)
PU: procesna enota (Processing Unit)
OQ: izhodna vrsta (Output Queue)
AG&FC: generator naslovov (Address Generator) in krmilnik pretoka (Flow Controller)
RC: osveževalni krmilnik (Refresh Controller)

Slika 4: Arhitektura procesorja μPD7281.

3.2. Vhodni IC in izhodni OC krmilnik

32 bitni vhodni paket vstopi v IC v obliki, kot jo prikazuje Slika 5.



Slika 5: 32 bitni vhodno-izhodni paket.

IC primerja lastni naslov, ki mu je bil dodeljen ob resetu, z MN poljem vhodnega paketa. Če se naslova ne ujemata, se tuji paket takoj prenese v OC. Pakete takšnih oblik imenujemo PASS paketi. Če pa se naslova ujemata, se MN polje izloči in tako spremenjen paket pošlje v LT. IC hkrati ugotavlja, če je v procesorju prostor za novi paket in ga po potrebi zadrži. OC pošilja 32 bitne izhodne pakete, katerih oblika je enaka vhodnim paketom (Slika 5). Izhodni paketi so bodisi podatkovni paketi iz OQ, statusni paketi, ki jih generira OC v primeru napak, dump paketi ali tuji vhodni paketi, dobljeni iz IC, ki se tu imenujejo PASSO paketi.

Tuji paket na vhodu (vh. paket PASS):

MN'	0	ID	CTRL	podatki

Tuji paket na izhodu (izh. paket PASSO):

MN'	0	ID	CTRL	podatki

Opomba: MN' se ne ujema z MN danega procesorja.

3.2.1. Načini delovanja:

Procesor lahko deluje v treh načinih: normalnem (Normal), testnem (Test) in prekinitvenem (Break) načinu. Po hardverskem resetu je procesor v normalnem načinu delovanja. V tem načinu poteka vpis, branje in izvrševanje podgrafa. V testnem načinu delovanja je omogočeno testiranje izvrševanja. Procesor preide v testni način, ko sprejme vhodni paket SETBRK. Če med delovanjem pride do nasičenja vrste DQ ali GQ, preide procesor v prekinitveno delovanje, opisano z vhodnim paketom SETMD. V prekinitveni način lahko preide procesor tudi iz testnega načina, tako da dobi paket CBRK. V prekinitvenem načinu delovanja je omogočeno dumpanje. Iz obeh načinov se vrne v normalni način po sprejetju vhodnega paketa CRESET.

Prehod v testni način (vh. paket SETBRK):

MN	0	ID	01101	M(1) Count(15)

Opomba: M=1 prekini izvrševanje po Count ciklih
M=0 prekini po Count dostopih do LT lokacije, naslovljene z ID

Prehod v prekinitveni način (vh. paket CBRK):

0000	0	01001

Vrsta prekinitvenega načina (vh. paket SETMD):

MN	0	01011	par. za prek. način

Opomba: Parametri za prekinitveni način določajo stopnjo vhodnih omejitev ter njihovo trajanje.

Sporočilo o napaki (izh. paket ERR):

0000	0	0000000	01001	MN(4) MODE(4) OOOSY(5)

Programski reset (vh. paket CRESET):

MN	0	01000

Opomba: Programski reset povzroči le nadaljevanje izvrševanja v normalnem načinu, za razliko od hardverskega, ki resetira tabele ter dodeli naslove procesorjem.

3.2.2. Vpis podgrafa:

Predhodni vpis podgrafa poteka s pomočjo vhodnih paketov SETLT, SETFTR, SETFTL in SETFTT.

Vpis povezave v LT (vh. paket SETLT):

MN	0	adr. v LT	11000	podatki za v LT

Vpis točke v FTR (vh. paket SETFTR):

MN	0	adr. v FT	11010	podatki za v FTR

Vpis točke v FTL (vh. paket SETFTL):

MN	0	adr. v FT	11101	podatki za v FTL

Vpis točke v FTT (vh. paket SETFTT):

MN	0	adr. v FT	11111	podatki za v FTT

3.2.3. Izpis podgrafa:

Vsebino LT in FT lahko beremo s pomočjo paketov RDLT, RDFTR, RDFTL in RDFTT, ki pomenijo zahtevo po branju. Prebrana vsebina lokacij v LT in FT pa se nahaja v izhodnih paketih LTRDD, FTRRDD, FTLRDD in FTTRDD.

Zahteva za branje iz LT (vh. paket RDLT):

MN	0	adr. v LT	10001

Prebrani podatki iz LT (izh. paket LTRDD):

0000	0	adr. v LT	11000	podatki iz LT

Zahteva za branje iz FTR (vh. paket RDFTR):

MN	0	adr. v FT	11001

Prebrani podatki iz FTR (izh. paket FTRRDD):

0000	0	adr. v FT	11001	podatki iz FTR

Zahteva za branje iz FTL (vh. paket RDFTL):

MN	0	adr. v FT	11010

Prebrani podatki iz FTL (izh. paket FTLRDD):

0000	0	adr. v FT	11010	podatki iz FTL

Zahteva za branje iz FTT (vh. paket RDFTT):

MN	0	adr. v FT	11011

Prebrani podatki iz FTT (izh. paket FTTRDD):

0000	0	adr. v FT	11011	podatki iz FTT

3.2.4. Izvrševanje podgrafa:

Ko je podgraf vpisan v procesor, se lahko prične njegovo izvrševanje - pretok podatkov po podgrafu. Procesor dobi vhodne podatke (operande) s pomočjo vhodnih paketov EXEC, rezultate pa vrne s paketi OUTD.

Vhodni podatki (vh. paket EXEC):

MN	0	ID	00CS1	operand

Izhodni podatki (izh. paket OUTD):

MN	0	ID	00CS1	rezultat

3.2.5. Dumpanje:

V prekinitvenem načinu delovanja je omogočeno tudi opazovanje vsebine delov paketov. V ta namen se uporabljata vhodni paket DUMP ter pripadajoči izhodni paket DUMPD.

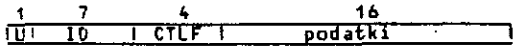
Zahteva za izpis danega polja (vh. paket DUMP):

MN	0	x(4)	DUMP(3)	10111

```
Izpisana vsebina polja (izh. paket DUMPD):
|0000|010(4)DUMP(3)|011111|   dumpani podatki   |
Opomba: glede na bite DUMP(3) dobimo:
        DUMP(3)      dumpani podatki
         000    x(5) GQ(5) DQ(6)
         001    x(4) u(1) ID(7) CTLF(4)
         010    DATA(16)
         011    x(3) u ID(7) x C S C S
         100    xx FTL(spodnjih 12) xx
         101    DATA (16)
         110    DATA (16)
         111    x(9) ID(7)
```

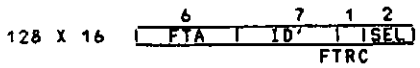
3.3. Tabela povezav LT

LT je 128 X 16 bitni dinamični RAM. V LT vstopi 28 bitni paket kot ga prikazuje Slika 6.



Slika 6: 28 bitni LT paket.

Identifikator LT paketa, ki ga pošlje IC, služi za naslovitev lokacije v LT. Vsebina naslovljene lokacije vsebuje 6 bitni naslov lokacije v tabeli točk (FTA), 7 bitni identifikator (ID'), FTRC bit in 2 bitno selekcijsko polje (SEL). Prikazana je na Sliki 7.



Slika 7: Vsebina lokacije v LT.

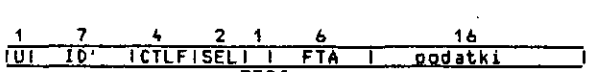
ID LT paketa se nadomesti z novim ID', vsebovanim v naslovljeni lokaciji v LT. Torej vsakokrat, ko paket preide čez LT, dobi novi identifikator. LT paketu se dodajo še FTA, FTRC in SEL polje. FTA polje služi za dostop do lokacije v FT. FTRC bit in SEL polje pa služita pri določanju tipa ukaza.

SEL	Tip ukazov
11	AG/FC
01	PU
10	GE
00	OUT

Opomba: Tipi ukazov so opisani v poglavju 3.11.

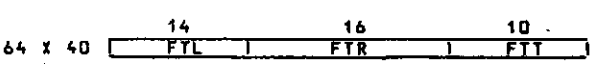
3.4. Tabela točk FT

FT je 64 X 40 bitni dinamični RAM. Vanj vstopijo paketi, ki jih pošlje LT v obliki, kot jo prikazuje Slika 8.



Slika 8: 37 bitni FT paket.

FTA polje je naslov lokacije v FT, katere vsebino sestavljajo 14 bitno polje FTL, 16 bitno polje FTR in 10 bitno polje FTT, ki vsebujejo krmilne podatke o različnih tipih ukazov. FT lokacijo prikazuje Slika 9.



Slika 9: Vsebina lokacije FT.

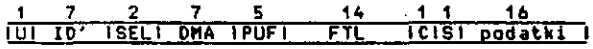
SEL polje FT paketa določa tip ukaza. Ukazi so lahko tipa OUT, GE, PU ali AG/FC. Ukazi tipa OUT narekujejo neposreden prenos paketa v OQ. Ukazi tipa GE se uporabljajo za generiranje paketov znotraj procesorja. PU ukazi omogočajo aritmetično logične operacije, AG/FC ukazi pa služijo AG&FC enoti pri delu z DM. Pri ukazih tipa PU se uporablja FTL polje, pri ostalih ukazih pa tudi FTR in FTT polje.

3.5. Generator naslovov in krmilnik pretoka AG&FC

AG&FC generira naslove lokacij v DM ter krmili njihovo branje in vpisovanje. AG&FC ugotavlja, če prispeli paket vsebuje ukaze z enim ali dvema operandoma. Če potrebuje en operand, se paket prenese direktno v Q. Če pa potrebuje ukaz dva operanda, morata biti na voljo oba, da se lahko prenese v Q. Paket, ki vsebuje prevega izmed prispelih operandov, se začasno shrani v DM, dokler ne prispe paket z drugim operandom. Ko paket z drugim operandom izstopi iz FT, AG&FC generira naslov lokacije v DM, v kateri je shranjen paket s prvim operandom in pošlje oba v Q.

3.6. Paketni pomnilnik DM

DM je 512 X 18 bitni dinamični RAM, ki služi za hranjenje prvega od prispelih operandov ukaza. Paket, ki je nosilec drugega operanda, vstopi v DM v obliki prikazani na Sliki 10.



Slika 10: 54 bitni DM paket.

Polje DMA vsebuje naslov lokacije v DM v kateri je shranjen prvi operand. Podatki o ukazu namenjenemu PU se nahajajo v FTL. DM se uporablja tudi za začasno hranjenje drugih, npr. vhodno-izhodnih podatkov.

3.7. Vrsta Q

Q je 48 X 60 bitni dinamični RAM, ki služi kot FIFO pomnilnik. Paketi, ki vstopajo v Q, nastanejo iz DM paketa tako, da se DMA polje izloči in nadomesti z vsebino lokacije DM na katero je prej kazalo polje DMA (prvi operand). Q začasno hrani pakete, namenjene v PU ali OQ. Q je razdeljen v dva FIFO pomnilnika: 32 X 60 bitno podatkovno vrsto (DQ) in 16 X 60 bitno generatorsko vrsto (GQ). DQ je namenjen ukazom tipa PU, OUT in AG/FC in hrani pakete, ki so namenjeni v PU ali OQ. Vrsta GQ pa je namenjena le ukazom tipa GE, ki generirajo nove pakete. Vrsta DQ zadržuje izhodne pakete, če je izhodna vrsta OQ polna. Podobno DQ zadržuje pakete, namenjene v PU, če je le-ta zasedena. Nekritično pošiljanje paketov v krožni obtok bi lahko privedlo do nasičenja vrste Q, zato je delovanje Q omejeno s sledečo zahtevo: če se v DQ nahaja osem ali več paketov, je branje iz GQ onemogočeno, če pa je v DQ manj kot osem paketov, ima branje iz GQ višjo prioriteto od branja iz DQ. Do nasičenja vrste Q bi lahko prišlo še v primeru, ko bi bila hitrost procesiranja manjša od hitrosti prihajanja novih vhodnih paketov v procesor. Zato v primeru, ko se v DQ nahaja več kot 23 paketov, procesor preide v prekinitveni način delovanja, s čimer se izogne nasičenju Q.

3.8. Izhodna vrsta OQ

OQ je 8 X 32 bitni statični RAM, organiziran kot FIFO pomnilnik. Služi začasnemu shranjevanju izhodnih paketov, prispelih iz DQ, ki se nato preko OC pošljejo na izhodno podatkovno vodilo. Če je OQ poln, pošlje ustrezen signal v DG, ter preneha sprejemati pakete. OQ paket se ujema s Q paketom.

3.9. Procesna enota PU

PU izvršuje ukaze tipa PU in GE. Koda PU ukaza se nahaja v polju FTL PU paketa. Ukazi tipa GE se uporabljajo za generiranje novih paketov, kopiranje posameznih ali skupin paketov in spreminjanja vsebine CTLF polja. Če potrebuje PU za izvršitev ukaza več kot en cikel, pošlje signal vrsti Q in IC, ki zadrži njuno delovanje. PU paket je po obliki enak Q paketu.

3.10. Osveževalni krmilnik RC

RC avtomatično generira pakete za osveževanje vseh dinamičnih RAMov v procesorju. Paket, ki ga generira RC, vstopi v IC, ter nato po vrsti še v LT, FT, DM in Q in vsakokrat povzroči osvežitev ustreznega RAMa. Po prihodu v Q se osvežitveni paket uniči.

3.11. Nabor ukazov

Kot smo omenili v poglavju 3.4. obstajajo štirje tipi ukazov: AG/FC, PU, GE in OUT.

3.11.1. Ukazi tipa AG/FC:

Tip AG/FC vsebuje 16 ukazov, ki jih razdelimo v tri skupine: AG, FC in AG/FC tip. Ukazi tipa AG so: RDCYCS, RDCYCL, WRCYCS, WRCYCL, RDWR in RDIDX. Ukazi tipa FC so: PICKUP, COUNT, CUT, DIVCYC, DIV, DIST, CONVO, SAVE in CNTGE. Ukaz QUEUE pa je tipa AG/FC.

Ukazi tipa AG/FC so opisani v FTR polju tabele FT, kjer zgornji štirje biti predstavljajo operacijsko kodo. Ostali del polja FTR in polje FTT pa vsebujejo ostale parametre AG/FC ukazov. Pomeni ukazov tipa AG/FC so:

- QUEUE: shrani prvi prispeli operand dvomestnega ukaza v DM.
- RDCYCS: ciklično branje podatkov iz izbranega področja v DM, kjer je največja dolžina področja 16 lokacij.
- RDCYCL: enako kot RDCYCS, le da je največja dolžina področja 256 lokacij.
- WRCYCS: ciklično vpisovanje podatkov v izbrano področje v DM, kjer je največja dolžina področja 16 lokacij.
- WRCYCL: enako kot WRCYCS, le da je največja dolžina področja 256 lokacij.
- RDWR: branje ali vpisovanje v DM. Z bitom FTRC izbiramo bodisi branje ali vpis.
- RDIDX: branje iz DM.
- PICKUP: izloči vsak n-ti paket in mu priredi ID + 1.
- COUNT: podvoji vsak n-ti paket in mu priredi ID + 1.
- DIVCYC: na vsakih n paketov izloči prvih m med njimi in jim priredi ID + 1.
- DIV: po vsakem prihodu paketa s FTRC = 1 se naslednjim n paketom ID ne spremeni, ostalim pa se priredi ID + 1 (paket s FTRC = 1 se uniči).
- DIST: zaporedje vhodnih paketov vsebuje pakete s FTRC = 0 ali 1. Po prihodu j-tega paketa s FTRC = 1 se vsem nadaljnim paketom, ki imajo FTRC = 0 in naslednjemu paketu s FTRC = 1, priredi ID + (j MOD k).
- SAVE: uporablja se pri nastavljanju ID polja.
- CUT: po vsakem prihodu paketa s FTRC = 1 se se ta paket skupaj z naslednjimi n paketi uniči, ostali pa ostanejo nespremenjeni.
- CONVO: uporablja se za komulativno računanje, npr. vsot in produktov.
- CNTGE: uporablja se pri generiranju več kot 16 kopij danega paketa (skupaj z ukazom COPYBK tipa GE, ki je opisan spodaj).

Konstante k, m in n so podane v FTT polju.

3.11.2. Ukazi tipa PU:

Ukazi tipa PU so shranjeni v FTL polju tabele FT. Uporabnih je le 12 spodnjih bitov, med katerimi so biti 0 do 4 operacijska koda ukaza, preostali biti pa se uporabljajo za dodatno informacijo o številu operandov (eden ali dva), legi operandov pri nekomutativnih operacijah, številu rezultatov (eden ali dva) ter tipu primerjave, podanim z biti PNZ (EQ, LT, LE, GT, GE in NE). Ukazi tipa PU so:

- Logični: OR, AND, EXOR, ANDNOT in NOT.
- Aritmetični: ADD, SUB, MUL, INC, DEC, NOP in ADDSC, SUBSC, MULSC ter NOPSC. Ukazi *SC najprej izračunajo operacijo * in vrnejo lego skrajnega levega setiranega bita.
- Premikalni: SHL in SHR ter SHLBRV in SHRBRV, kjer ukaza *BRV predhodno obrneta vrstni red bitov.
- Primerjalni: CMPNOM, CMP in CMPXCH. Vhodna operanda A in B se primerjata na način podan s PNZ poljem. Rezultata primerjave sta X in Y, kot sledi

			C_x	S_x	X	C_y	S_y	Y
CMPNOM	pri	PNZ=FALSE	0	0	0000H	0	0	0000H
	pri	PNZ=TRUE	1	0	0001H	1	0	0000H
CMP	pri	PNZ=FALSE	0	S_a	A	0	S_b	B
	pri	PNZ=TRUE	1	S_a	A	1	S_b	B
CMPXCH	pri	PNZ=TRUE	C_a	S_a	A	C_b	S_b	B
	pri	PNZ=FALSE	C_b	S_b	A	C_a	S_b	A

- Operacije nad biti: GET1, SET1 in CLR1.
- Testiranje bitov: ANDMSK in ORMSK.
- Pretvorba podatka: CVT2AB in CVTAB2. CVT2AB pretvori 16 bitni operand, zapisan v sistemu z dvojiškim komlementom, v 17 bitno besedo v sistemu predznak-absolutna vrednost. Predznak se nahaja v bitu S . Drugi ukaz je inverzen prvemu. Ob prekoračitvi se postavi bit C .
- Popravki pri dvojni natančnosti: ADJL. Če imata besedi, s katerima je zapisan večnatančnostni operand, različna predznaka, ju ADJL popravi.
- Kumulativno seštevanje: ACC. Ukaz povzroči seštetje podatkovnih polj v zaporednju paketov. Vsebina vsakega prispelega paketa se prišteje k delni vsoti v registru ACC. Paketi v zaporedju so bodisi tipa 1 ali 2. Paket tipa 1 se po prištetju svoje vsebine zbriše, paket tipa 2 pa po prištetju svoje vsebine prevzame vsebino registra ACC.
- Kopiranje kontrolnega bita : COPYC.

3.11.3. Ukazi tipa GE:

Ukazi tipa GE so opisani v polju FTL tabele FT. Operacijsko kodo sestavljata dva bita, preostali biti pa določajo: če je treba operanda pred vstopom v Q zamenjati, število operandov ter število kopij pri generiranju paketov. V to skupino sodijo trije ukazi:

- COPYBK: generiranje bloka kopij danega paketa. Vsi novi paketi imajo enak ID kot originalni paket, le zadnji ima ID + 1. Vrednosti v podatkovnih poljih se lahko povečujejo ali zmanjšujejo za konstantno vrednost.
- COPYM: generiranje skupine kopij danega paketa z različnimi ID. Podatkovno polje se lahko spreminja podobno kot pri ukazu COPYBK.
- SETCLT: branje ali spreminjanje vsebine tabel LT in FT. Ukaz SETCLT omogoča programe, ki se sami spreminjajo, saj se uporablja v času izvrševanja grafa.

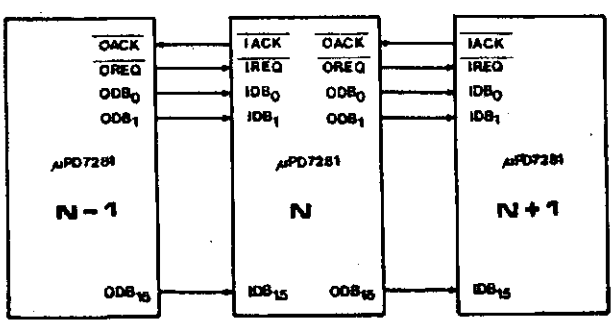
3.11.4. Ukazi tipa OUT:

Ukazi tipa OUT so opisani v FTL polju tabele FT. Polje vsebuje podatke o tem, če nastopa OUT ukaz samostojno ali skupaj z nekim AG/FC ukazom, če je treba operanda pred vstopom v OQ zamenjati, operacijsko kodo OUT ukaza ter naslov procesorja MN, kateremu je namenjen izhodni paket. Ločimo dva ukaza tipa OUT:

- OUT1: povzroči prenos 32 bitnega paketa na izhodno vodilo. Paket vsebuje podatek, njegov identifikator, ter naslov procesorja, kateremu je namenjen.
- OUT2: povzroči prenos 64 bitnega paketa na izhodno vodilo. Ukaz se uporablja pri pošiljanju števil z dvojno natančnostjo.

3.12. Načini povezovanja μPD7281

Procesorji μPD7281 se povezujejo v večprocesorski sistem na dva osnovna načina: kaskadni in krožni. Pri kaskadnem povezovanju ni potrebna dodatna materialna oprema. Največ 14 čipov povezujemo neposredno, kot prikazuje Slika 11. Izmenjava vhodno-izhodnih paketov poteka s pomočjo signalov zahteva/potrditev (REQ/ACK).



Slika 11: Povezovanje procesorjev μPD7281.

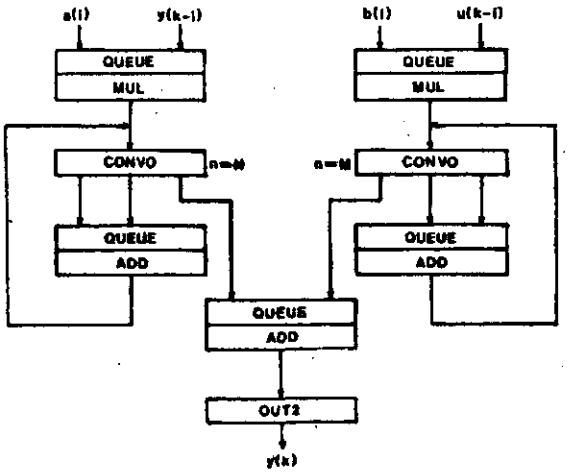
Za krožno arhitekturo je potrebna dodatna materialna oprema, zato je v razvoju podporni čip MAGIC (Memory Access and General bus Interface Chip).

3.13. Programiranje μPD7281

Bogat nabor ukazov omogoča zelo učinkovito programiranje problemov s področja obdelave signalov ter zahtevnega numeričnega računanja. Za ilustracijo si oglejmo realizacijo linearnega digitalnega filtra [5], podanega z enačbo

$$y(k) = \sum_{i=1}^{N} a(i)y(k-i) + \sum_{i=1}^{M} b(i)u(k-i),$$

kjer je u(k) vhodni in y(k) izhodni signal. Pripadajoči program (graf), zapisan v 'strojnem' jeziku, prikazuje Slika 12.



Slika 12: Rekurzivni izračun digital. filtra.

Seveda je razvit tudi zbirni jezik, ki omogoča lažji opis programskega podgrafa [23]. Tudi uporabo zbirnika ilustrirajmo na primeru linearnega digitalnega filtra, tokrat realiziranega v prostoru stanj, kot ga podajata enačbi

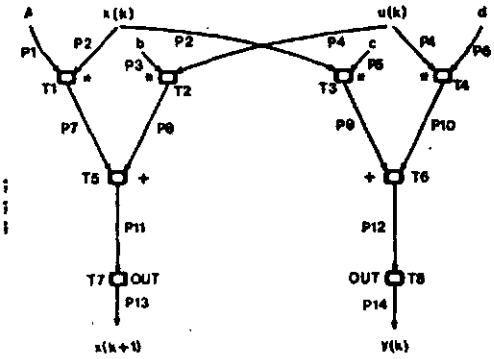
$$x(k+1) = Ax(k) + bu(k)$$
$$y(k) = cx(k) + du(k),$$

kjer so A, b, c in d usrezne matrike in vektorji. Program bi se v primeru, ko so matrike in vektorji enodimenzionalni, glasil:

```
EQUATE    HOST  = 0;
MODULE    PROC1 = 1;
INPUT     P1,P2,P3,P4,P5,P6;
OUTPUT    P13,P14;
LINK      P7  = T1 (P1, P2);
LINK      P8  = T2 (P3, P4);
LINK      P9  = T3 (P2, P5);
LINK      P10 = T4 (P4, P6);
LINK      P11 = T5 (P7, P8);
LINK      P12 = T6 (P9, P10);
LINK      P13 = T7 (P11,  );
LINK      P14 = T8 (P12,  );
FUNCTION T1 = MUL,QUEUE(Q1,1);
FUNCTION T2 = MUL,QUEUE(Q2,1);
FUNCTION T3 = MUL,QUEUE(Q3,1);
FUNCTION T4 = MUL,QUEUE(Q4,1);
FUNCTION T5 = ADD,QUEUE(Q5,1);
FUNCTION T6 = ADD,QUEUE(Q6,1);
FUNCTION T7 = OUT1(HOST,0);
FUNCTION T8 = OUT1(HOST,0);
MEMORY    Q1 = AREA(1);
MEMORY    Q2 = AREA(1);
MEMORY    Q3 = AREA(1);
MEMORY    Q4 = AREA(1);
MEMORY    Q5 = AREA(1);
MEMORY    Q6 = AREA(1);
```

Zgornji program se izvaja samo v procesorju PROC1, čeprav je v algoritmu prisotna določena stopnja vzporednosti. Težava je v tem, da je potencialna vzporednost slabo razvidna, saj je to le enodimenzionalni zapis programskega (pod)grafa. Zato običajno poteka programiranje tako, da najprej konstruiramo programski graf, določimo vzporedno izvršljive podgrafe ter jih zapišemo v zbirniku. Zgornji program izhaja iz grafa, ki je podan na Sliki 13. Opazimo, da se lahko levi (izračun x(k+1)) in desni (izračun y(k)) del grafa izvršujeta vzporedno, zato levi del priredimo procesorju PROC1, desni del pa procesorju PROC2. Ustrezna programa, ki sta v tem primeru zelo podobna, sta podana na Sliki 13. Razlika med njima je le v tem, da PROC1 pošilja rezultat procesorju PROC2, medtem ko procesor PROC2 pošilja rezultat glavnemu procesorju HOST.

```
EQUATE    PROC2 = 2;
MODULE    PROC1 = 1;
INPUT     P1,P2,P3,P4;
OUTPUT    P13;
LINK      P7  = T1 (P1, P2);
LINK      P8  = T2 (P3, P4);
LINK      P11 = T5 (P7, P8);
LINK      P13 = T7 (P11,  );
FUNCTION  T1 = MUL,QUEUE(Q1,1);
FUNCTION  T2 = MUL,QUEUE(Q2,1);
FUNCTION  T5 = ADD,QUEUE(Q5,1);
FUNCTION  T7 = OUT1(PROC2,0);
MEMORY    Q1 = AREA(1);
MEMORY    Q2 = AREA(1);
MEMORY    Q5 = AREA(1);
```




```
EQUATE    HOST  = 0;
MODULE    PROC2 = 2;
INPUT     P2,P4,P5,P6;
OUTPUT    P14;
LINK      P9  = T3 (P2, P5);
LINK      P10 = T4 (P4, P6);
LINK      P12 = T6 (P9, P10);
LINK      P14 = T8 (P12,  );
FUNCTION  T3 = MUL,QUEUE(Q3,1);
FUNCTION  T4 = MUL,QUEUE(Q4,1);
FUNCTION  T6 = ADD,QUEUE(Q6,1);
FUNCTION  T8 = OUT1(HOST,0);
MEMORY    Q3 = AREA(1);
MEMORY    Q4 = AREA(1);
MEMORY    Q6 = AREA(1);
```

Slika 13: Realizacija digitalnega filtra v prostoru stanj.

Pri večini programskih grafov je določanje vzporedno izvršljivih podgrafov zahtevno, kar narekuje razvoj metod za avtomatično iskanje vzporednosti ter optimizacijo glede na število uporabljenih procesorjev [4]. Zelo dobrodošel bi bil grafični zbirnik za generiranje kode neposredno iz programskega grafa ter pascalski ali C prevajalnik za generiranje in optimizacijo programskih grafov.

4. ZAKLJUČEK

Rezultati testov opravičujejo uporabo večprocesorske arhitekture z μPD7281 [1]. Tako npr. rotacija binarne slike velikosti 512 X 512 točk zahteva 0.6s pri krožni povezavi treh procesorjev; en procesor pa potrebuje 1.5s. Za izračun funkcije cos(x) potrebuje en procesor 40μs, kaskada treh procesorjev pa 15μs. V splošnem se čas obdelave eksponentno zmanjšuje z večanjem števila uporabljenih procesorjev, žal pa ne neomejeno, saj se pri večjem številu procesorjev pojavijo zastoji pri pretoku vhodno-izhodnih paketov med procesorji. Zato je odvisno od same aplikacije, ali jo bomo razbili glede na podatke ali na programski graf. Vzemimo npr. obdelavo slike, ki obsega naloge kot so premik, normiranje, rotacija itd. Če se odločimo za razbitje glede na podatke, razbijemo sliko na podslike, tako da se vsaka podslika istočasno obdela v svojem procesorju. To pomeni, da je v vseh procesorjih cel programski graf, vanj pa vstopajo le podatki o pripadajoči podsliki. Razbitje glede na graf (program) pa pomeni, da prvi procesor opravi premik slike, drugi normiranje, tretji rotacijo itd. To pomeni, da je v vsakem procesorju drug podgraf in vanj vstopajo podatki o celi sliki. Seveda pa je razbitje grafa smiselno le tedaj, ko se lahko podgrafi izvršujejo vzporedno.

5. LITERATURA

[1] Chong Y.M.: Data Flow Chip Optimizes Image Processing, Computer Design, Oct. 15, 1984, pp.97-103

[2] Jeffery T.: The μPD7281 Processor, Byte, November 1985, pp.237-246

[3] μPD7281 Image Pipelined Processor, NEC Electronics, February 1985

[4] Robič B., J.Šilc: On Choosing a Plan for the Execution of Data Flow Program Graph, Informatica 3/86

[5] Ohba N., T.Saito, Y.Hoshiko: Signal Processing on a Data-Flow Processor, Microprocessing and Microprogramming 14,1984, pp.17-27