

# PATENTNA PRIJAVA

**Naslov izuma:**

**ARITMETIČNI DELILNIK S PORAVNAVANJEM**

**Vlagatelj:**

Odsek za komunikacijske sisteme  
Inštitut Jožef Stefan  
Jamova 39, 1000 Ljubljana, Slovenija

**Izumitelja:**

- Rainer Trummer  
Department of Computer Science  
University of Salzburg  
Jakob-Haringer-Str. 2, 5020 Salzburg, Austria
- Roman Trobec  
Odsek za komunikacijske sisteme  
Inštitut Jožef Stefan  
Jamova 39, 1000 Ljubljana, Slovenija

**Predlagana recenzenta:**

- prof. dr. Franc Novak  
Odsek za računalniške arhitekture  
Inštitut Jožef Stefan  
Jamova 39, 1000 Ljubljana, Slovenia
- dr. Viktor Avbelj  
Odsek za komunikacijske sisteme  
Inštitut Jožef Stefan  
Jamova 39, 1000 Ljubljana, Slovenia

**Mesto in datum:**

Ljubljana, 15. november 2005

# ARITMETIČNI DELILNIK S PORAVNAVANJEM

## Opis izuma

### Področje izuma

Na podlagi primerjave s sorodnimi patenti sodi predlagani izum, po mednarodni patentni klasifikaciji, na področje G06F7.

### Področje tehnike

Izum obsega ožje področje aritmetičnih delilnikov celih števil in je del širšega področja vezji za izvedbo aritmetičnih operacij. Deljenje je ena od najzahtevnejših aritmetičnih operacij, ki v splošnem ne izračuna natančnega rezultata, saj deljenec ni nujno mnogokratnik delitelja. Količnik in ostanek dobimo običajno z zaporedjem operacij, ki teče toliko časa, da dosežemo zahtevano natančnost. Ta postopek imenujemo *zaporedno deljenje* in služi kot podlaga za mnoge praktične izvedbe delilnikov; od enostavnih, ki slonijo na številčni osnovi 2, do najzahtevnejših in hitrih vzporednih delilnikov.

Pričujoč izum spada v kategorijo tako imenovanih *odštej-in-pomakni* delilnikov, ki med postopkom deljenja izvajajo zaporedna odštevanja delitelja od deljenca ter pomike delitelja, podobno kot pri ročnem deljenju, dokler končno ne izračunajo količnik in ostanek deljenja. Ena od podskupin teh delilnikov se imenuje *podatkovno odvisni* delilniki, ki za izvedbo deljenja porabijo različen čas, odvisen od vrednosti deljenca in delitelja. Osnovna ideja, na kateri slonijo taki delitelji, je preskakovanje odvečnih računskih operacij z izvajanjem pomikov, dokler so prisotne ničle na bolj pomembnih mestih (vodilne ničle) v vmesnem ostanku ali delitelju. Ta metoda se imenuje *pomikanje preko ničel*.

### Prikaz problema

Visoko zmogljivi in hitri delilniki običajno zahtevajo za svojo izvedbo preveč prostora, da bi jih lahko vgradili v majhne elektronske naprave kot so signalni procesorji ali podobne vgrajene arhitekture. Na različnih področjih uporabe so potrebe po hitrih aritmetičnih enotah ves večje, zato je iskanje novih, izboljšanih rešitev za deljenje še vedno aktualno.

Poglavitna slabost podatkovno odvisnih delilnikov je v tem, da je čas, potreben za izvedbo deljenja, odvisen od vrednosti operandov, pri čemer je pri mnogih vrednostih le-teh čas deljenja krajši od časa, potrebnega na delilnikih s konstantnim časom deljenja. Predlagani delilnik izvaja deljenje v krajšem času, ker uporablja kombinatorično logično vezje, imenovano *poravnalnik*, ki omogoča zmanjšanje števila operacij, potrebnih za poljuben par operandov. Doseženo izboljšanje sloni na ideji ponavljajočega poravnavanja delitelja pred vsakim odštevanjem, zato da se prilagodi velikosti vmesnega ostanka. Spomnimo, da pomik binarnega števila v levo/desno za  $n$  mest ustreza množenju/deljenju tega števila z  $2^n$ . Posledično se bo tako v enem koraku od vmesnega ostanka odštela največja možna vrednost, ki predstavlja delitelj, pomnožen z ustrežno potenco števila 2, zaradi česar se bo zmanjšalo število potrebnih operacij.

## Stanje tehnike

Področje aritmetičnih operacij, še posebej deljenja, je zelo dobro raziskano in opisano v številnih knjigah ali znanstvenih člankih [1, 2, 3, 4]. Za njegovo izvedbo so bili razviti številni različni postopki. Najpreprostejši sloni na zaporednem deljenju z osnovo  $r$  [1], kjer  $r$  označuje osnovo (radix) številskega sistema, ki je tipično izbrana izmed potenc števila 2. Za izračun količnika potrebujejo taki delilniki konstantno število iteracij, ki je odvisno od uporabljene številske osnove. Za pospešitev zaporednega deljenja so razvili vrsto postopkov, med katerimi je najučinkovitejše tako imenovano *SRT* deljenje [5], ki predstavi delitelj  $D$  in vmesni ostanek  $R_i$  kot normalizirani števili, ki ustrezata pogojema  $1/2 < |D| < 1$  in  $1/2 < r|R_i| < 1$ . Ob uporabi *redundantnih digitov*, se določeno število najpomembnejših bitov delitelja in vmesnega ostanka uporabi za izbiro ustreznega tekočega digita količnika, ki ga najdemo v pomnilniški (look-up) tabeli. Najpomembnejša slabost SRT deljenja je ravno v tem, da potrebuje veliko pomnilniško tabelo, katere velikost je kvadratično odvisna od velikosti številske baze.

Pomemben del SRT deljenja je tudi normaliziranje delitelja in vseh vmesnih ostankov. Izveden je z metodo, imenovano *premikanje preko ničel*, ki je podobna metodi uporabljeni v poravnalniku predlaganega izuma. Vendar je to metoda s premikanjem preko ničel izvedena z  $2n$ -bitnim pomikalnim registrom, ki zahteva poseben postopek za ugotavljanje števila vodilnih ničel, poleg tega pa ne generira niti vmesnih količnikov, niti ne katerega koli od bitov končnega rezultata deljenja [2].

Področje aritmetičnih delilnikov pokriva okrog sto patentov iz ameriških ali evropskih baz. V nadaljevanju omenjamo le nekatere od njih, ki so najbolj povezani s pričujočim izumom.

V patentu US-5014233 [6] je predlagan delilnik, ki sloni na iterativni *odštej-in-premakni* metodi, podobno kot delilnik, predlagan v pričujoči inovaciji. Toda delilnik, predlagan v patentu US-5014233, zahteva konstantno število operacij, ki je sorazmerno s številom bitov deljenca, saj ne zazna ničel v operandih, zaradi česar je število potrebnih korakov za izračun kvocienta večje kot je potrebno.

V patentu US-5493522 [7] je opisana metoda za izvedbo hitrega deljenja po modulu 32, ki sloni na modularnem operatorju deljenja in kombinatorični logiki za izračun produkta med modularnim operatorjem in deljencem, kateremu sledi pomikanje za ustrezno število mest, da bi dobili končni rezultat kot število po modulu 32. Ta patent se nanaša na naš izum v tem smislu, da za določitev potrebnega števila mest v pomikalnem registru uporablja le kombinatorično logiko, ki pa je popolnoma drugačna od tiste, ki je uporabljena v predlaganem poravnalniku.

V patentu EP-1351129A2 [8], ki je glede na predlagan izum najrelevantnejši patent iz baze, je opisano deljenje dveh binarnih števil, deljenca in delitelja, ki izračuna kvocient in ostanek v iterativnem postopku s poravnavanjem najpomembnejših od nič različnih bitov (MSB) deljenca in delitelja, izvedenega s pomikanjem delitelja za relativno razliko med položajema MSB v obeh operandih, ki mu sledi množenje vmesnega rezultata z 2 ter primerjanje deljenca z deliteljem zaradi določitve zaustavitvenega kriterija. Relativna razlika med položajema obeh MSB se določi s kombinatoričnim vezjem, ki generira dve vrednosti, kateri označujeta položaj vsakega MSB. Z odštevanjem teh dveh vrednosti dobimo število mest, za katero je treba v začetku delitelj pomakniti v levo. Število potrebnih iteracij za izračun rezultata je tako vedno enako številu preostalih mest v delitelju. Začetni

pomik je uporabljen tudi v našem izumu, toda nadaljnji postopek, uporabljen v deljenju iz patenta EP-1351129A2, ne izkorišča možnosti za neprestano poravnavanje, ki je vgrajeno v poravnalniku iz našega izuma.

Izjavljamo, da po našem vedenju nobeden od pregledanih, nam dostopnih patentov s področja deljenja, ne vsebuje osnovnih idej, uporabljenih v aritmetičnem delilniku s poravnavanjem, ki uporablja poravnalnik in dva seštevalnika, tako kot je opisano v pričujoči patentni prijavi.

## Opis nove rešitve

V naslednjih poglavjih je najprej podrobneje opisan bistven gradnik predlaganega delilnika, *poravnalnik*, zatem pa je podana natančnejša zgradba celotnega *aritmetičnega delilnika s poravnavanjem*.

### Spisek slik, tabel in oznak

#### SLIKE

Slika 1: prikazuje shematični diagram kombinatorične enote, ki jo imenujemo *poravnalnik*.

Slika 2: prikazuje shematični diagram kodirnika prioritete.

Slika 3: prikazuje shematični diagram logičnega pomikalnika v desno.

Slika 4: prikazuje shematični diagram logičnega pomikalnika v levo.

Slika 5: prikazuje shematični diagram celotnega *aritmetičnega delilnika s poravnavanjem*.

#### TABELE

Tabela 1: podaja primer poravnalnega postopka za delitelj  $D = 9$  in ostanek  $R = 197$  v 8-bitni binarni aritmetiki.

Tabela 2: podaja celoten proces deljenja z neprestanim poravnavanjem v primeru:  $197/9 = 21$  z ostankom 8, v 8-bitni binarni aritmetiki.

#### OZNAKE

- Pomembnejši gradniki so na slikah označeni s številkami v krogcih. Iz besedila se sklicujemo nanje s temi številkami.
- Binarne aritmetične operacije so označene z indeksom 2.
- Aritmetična operacija množenja je v formalnih predstavitvah označeno z zvezdico \*.

### Opis poravnalnika

Na sliki 1 je prikazana blokovna shema poravnalnika. Dva kodirnika prioritete **1** in **2** pretvorita  $n$ -bitni ostanek  $R_i$  in  $n$ -bitni delitelj  $D$  v  $n$ -bitna signala  $R_e$  oziroma  $D_e$ , ki imata edini od nič različen bit na položaju najbolj pomembnega od nič različnega bita v njunem vhodnem signalu, ali povedano drugače: vsi manj pomembni zadnji biti, ki sledijo vodilni enici v vhodnem signalu, se postavijo na nič. Izhod kodirnika prioritete bomo v nadaljevanju imenovali *ena-1* signal. Blokovni diagram 4-bitnega kodirnika prioritete je prikazan na sliki Figure 2. Kodirnik se lahko razširi na potrebno število bitov z enostavno razširitvijo logičnih vezji. Označimo število ničel, ki sledijo vodilni enici v  $R_e$  in  $D_e$ , z  $m$  oziroma z  $j$ . Da bi določili relativni odmik med vodilnimi enicami v  $R_i$  in  $D$ , pomaknemo izhodni signal iz kodirnika ostanka **1**,  $R_e$ , v desno za število zadnjih ničel v izhodnem signalu kodirnika delitelja **2**,  $D_e$ . Pomik izvedemo z uporabo izhodnega signala

kodirnika delitelja **2**,  $D_e$ , ki ga uporabimo kot  $n$ -bitni ena-1 izbirni signal v vezju  $n$  internih multiplekserjev, iz katerih je zgrajen logični pomikalnik v desno **3**. Blokovni diagram 4-bitnega logičnega pomikalnika v desno je prikazan na sliki Figure 3. Pomikalnik se lahko razširi na potrebno število bitov z enostavno razširitvijo logičnih vezji. Izhod pomikalnika v desno **3** je v nadaljevanju pomemben zaradi dveh razlogov: prvič, z njim je predstavljen  $n$ -bitni vmesni količnik  $P_{i+1}$ , ki vsebuje  $i$ -ti bit končnega količnika že na svojem pravem mestu, in drugič, služi kot  $n$ -bitni ena-1 izbirni signal v vezju  $n$  internih multiplekserjev, iz katerih je zgrajen logični pomikalnik v levo **4**, ki naredi  $n$ -bit poravnan vmesni delitelj  $A_{i+1}$ , kateri zaseda enako število pomembnih bitov kot  $R_i$ . Blokovni diagram 4-bitnega logičnega pomikalnika v levo je prikazan na sliki Figure 4. Pomikalnik se lahko razširi na potrebno število bitov z enostavno razširitvijo logičnih vezji.  $A_{i+1}$  dobimo s pomikanjem  $D$  v levo za število zadnjih ničel v  $P_{i+1}$ , ki je v skladu s prejšnjimi pojasnili enako  $m - j$ . Opisani postopek izrazimo na formalni način takole:

$$\begin{aligned} \text{prioriteta} : \{0, \dots, 2^n - 1\} &\longrightarrow \{2^m : 0 \leq m < n\} \cup \{0\} \\ \text{prioriteta}(0, \dots, 0, 1, r_{m-1}, \dots, r_0)_2 &= R_e = (0, \dots, 0, 1, 0, \dots, 0)_2 = 2^m, \end{aligned}$$

$$\begin{aligned} P_{i+1} &= \text{prioriteta}(R_i) / \text{prioriteta}(D) = R_e / D_e = 2^{m-j}, \\ A_{i+1} &= D * P_{i+1}. \end{aligned}$$

Postopek poravnave lahko predstavimo z zaporedjem operacij, ki jih združimo v sledeče tri korake:

- 1p Kodiraj prioriteto vmesnega ostanka  $R_i$  in delitelja  $D$  s kodirnikoma prioritete **1** in **2**, ki generirata signala  $R_e$  in  $D_e$ . Oba signala vsebujeta le vodilni od nič različen bit v  $R_i$  oziroma  $D$ , kateremu sledijo same ničle.
- 2p Pomakni  $R_e$  za  $j$  bitov v desno, pri čemer je  $j$  število zadnjih ničel v  $D_e$ , z uporabo kontrolnega signala  $D_e$  v logičnem pomikalniku v desno **3**, ki generira vmesni količnik  $P_{i+1}$ .
- 3p Pomakni  $D$  za  $m - j$  bitov v levo, pri čemer je  $m - j$  število zadnjih ničel v vmesnem količniku  $P_{i+1}$ , z uporabo kontrolnega signala  $P_{i+1}$  v logičnem pomikalniku v levo **4**, ki generira poravnan delitelj  $A_{i+1}$ .

Iz povedanega sledi, da potrebuje  $A_{i+1}$  za svojo predstavitev enako število vodilnih bitov kot  $R_i$ , za vsak  $R_i$ , ki zadošča relaciji  $R_i \geq D$ . V obeh,  $R_i$  in  $D$ , je lahko desno od vodilne enice kakršna koli kombinacija bitov. Iz tega sledi, da je lahko  $A_{i+1}$  manjši, večji ali enak kot  $R_i$ . Po drugi strani pa je  $A_{i+1}$ , pomaknjen v desno za eno mesto, enak  $\frac{1}{2}A_{i+1}$  in zato vedno manjši kot  $R_i$ .

### Izvedbeni primer poravnalnika

Opisani postopek poravnavanja z izvedbenim primerom za delitelj  $D = 9$  in ostanek  $R = 197$  v 8-bitni binarni aritmetiki je predstavljen v tabeli 1. V prvem in četrtem stolpcu sta vhoda ali izhoda kodirnikov prioritete **1** oziroma **2**.

Tabela 1: Primer poravnavanja delitelja  $D = (00001001)_2$  z ostankom  $R = (11000101)_2$  ter generiranja vmesnega količnika  $P = (00010000)_2$  s poravnanim deliteljem  $A = (10010000)_2$ .

R ali $R_e$	P	A	D ali $D_e$	opis	korak
11000101	.....	.....	00001001	priključena vhoda R in D	
10000000			00001000	kodiraj prioriteto: generiraj $R_e$ in $D_e$	1p
			00001000	signal $D_e$ vsebuje $j = 3$ zadnje ničle	
	00010000			pomakni $R_e$ v desno za $j$ bitov: generiraj P	2p
	00010000			signal P vsebuje $m - j = 4$ zadnjih ničel	
		10010000		pomakni D v levo za $m - j$ bitov: generiraj A	3p
00010000		10010000		vmesni količnik in poravnan delitelj	

### Opis aritmetičnega delilnika s poravnavanjem

Na sliki 5 je prikazana blokovna shema celotnega delilnika. Register za ostanek **5** inicializiramo tako, da vanj vpišemo dani  $n$ -bitni pozitivni deljenec  $R_0$ . Med postopkom deljenja shranjuje ta register vmesne ostanke  $R_i$ , in po končanem deljenju vsebuje končni ostanek  $R_k$ , ki je del rezultata. Register za delitelj **10** inicializiramo z danim  $n$ -bitnim pozitivnim deliteljem  $D$ . Med celotnim postopkom deljenja se vsebina tega registra ne spreminja. Register za količnik **15** inicializiramo tako, da vanj vpišemo  $n$  ničel začetnega količnika  $Q_0$ . Med postopkom deljenja shranjuje ta register vmesne vrednosti količnika  $Q_i$ , po končanem deljenju pa končni količnik  $Q_k$ , ki je del rezultata.

Naj opomnimo, da slika 5 ne vsebuje vezja, ki bi bilo potrebno za detekcijo izjemnega primera,  $D = 0$ , izvedenega na primer z  $n$ -vhodnimi NOR-vrati, povezanimi s pripadajočimi  $n$ -biti registra za delitelj **10**, ker smo predpostavili, da je testiranje primera  $D = 0$  narejeno pred začetkom postopka deljenja. Naj opomnimo še, da smo zaradi enostavnosti izpustili v sliki 5 tudi kontrolno vezje, izvedeno na primer s končnim avtomatom, ki generira nujno potrebne krmilne signale *clock* in *write* za registra ostanka **5** in količnika **15**.

Primerjalnik **6** preverja, če je izpolnjena relacija  $R_i < D$ ; če je, potem je postopek deljenja končan. V posebnih primerih, ko je  $R_0 < D$ , delilnik takoj preneha z deljenjem, s tem da register za ostanek **5** in za količnik **15** ostaneta nespremenjena, torej je končni količnik  $Q_k = Q_0 = 0$  in končni ostanek  $R_k = R_0 = D$ . V preostalih primerih poravnalnik **11** zaporedno generira  $n$ -bitne poravnane delitelje  $A_{i+1}$  in pripadajoče  $n$ -bitne vmesne količnike  $P_{i+1}$ , po postopku, ki je bil opisan v prejšnjem poglavju. Naj poudarimo, da sta obe vrednosti  $A_{i+1}$  in  $P_{i+1}$  generirani samo s kombinatorično logiko, zato sta začetni izhodni stanji  $A_0$  in  $P_0$  nedefinirani.

Inverter **12** pripravi izhod poravnane delitelja iz poravnalnika **11** za seštevanje z dvojiškim komplementom (two's complement addition), s tem da invertira  $A_{i+1}$ . Dva vzporedna seštevalnika **7** in **8** sta oba krmiljena s konstantnim vhodnim signalom prenosa (carry-in)  $PV=1$ , kar zagotavlja, da seštevalnika vedno izvajata le operacijo odštevanja. Zato bomo v nadaljnjem besedilu govorili, da s seštevalnikom izvajamo odštevanje. Iz prejšnjega poglavja vemo, da odštevanje  $A_{i+1}$  od  $R_i$  lahko povzroči prekoračitev, kar bo

vidno v negativnem rezultatu oziroma v od nič različni vrednosti izhodnega signala prenosa PI (carry-out). Po drugi strani pa odštevanje  $A_{i+1}$ , ki je pomaknjen za eno mesto v desno, ne more nikoli prekoračiti. Prav zato vsebuje predlagani delilnik dva seštevalnika, ki delujeta hkrati. Seštevalnik **7** dobi na vhod, ki ustreza drugemu členu, vseh  $n$ -bitov invertiranega  $A_{i+1}$ , medtem ko seštevalnik **8** dobi na enak vhod le  $n - 1$  vodilnih bitov invertiranega  $A_{i+1}$ , ki so dobljeni z ustrežno prevezavo vseh njegovih bitov za eno mesto v desno, kar ustreza operaciji  $\frac{1}{2}A_{i+1}$ . Izhodni signal prenosa PI iz seštevalnika **7** kontrolira dva multiplekserja **9** in **13**. Če odštevanje  $R_i - A_{i+1}$  izvedeno s seštevalnikom **7** ni prekoračeno, kar pomeni, da je rezultat pozitiven, potem bo multiplekser **9** izbral ta rezultat, multiplekser **13** pa  $P_{i+1}$ . Če bo odštevanje na seštevalniku **7** prekoračeno (negativni rezultat), bo njegov izhodni signal prenosa PI povzročil, da bo multiplekser **9** izbral rezultat odštevanja  $R_i - \frac{1}{2}A_{i+1}$ , ki ga je izvedel seštevalnik **8**, multiplekser **13** pa bo izbral  $\frac{1}{2}P_{i+1}$ . V obeh primerih bo izhod multiplekserja **9** shranjen kot  $R_{i+1}$  v registru za ostanek **5**, izhod multiplekserja **13** pa bo z OR-vrati **14** kombiniran s prejšnjim vmesnim količnikom  $Q_i$  in shranjen kot vmesni količnik  $Q_{i+1}$  v registru za količnik **15**.

Predstavimo gornjo razlago za generiranje vmesnega ostanka in količnika še formalno, s pravili, ki so navedena spodaj:

$$A_{i+1} = 2^{m-j} * D \quad \text{in} \quad \frac{1}{2}A_{i+1} = 2^{m-j-1} * D.$$

$$R_{i+1} = \begin{cases} R_i - 2^{m-j-1} * D & \text{if } R_i < 2^{m-j} * D \\ R_i - 2^{m-j} * D & \text{if } R_i \geq 2^{m-j} * D, \end{cases}$$

$$Q_{i+1} = \begin{cases} Q_i + 2^{m-j-1} & \text{if } R_i < 2^{m-j} * D \\ Q_i + 2^{m-j} & \text{if } R_i \geq 2^{m-j} * D, \end{cases}$$

za vse  $j, m \in \{0, 1, \dots, n - 1\}$ ,  $j \leq m$ .

Zaporedje operacij, ki jih je treba izvesti pri opisanem postopku *deljenja z neprestanim poravnavanjem*, lahko opišemo z naslednjimi koraki:

- 1d Naloži register za ostanek **5** z deljencem  $R_0$ , naloži register za delitelj **10** z deliteljem  $D$  ter naloži register za količnik **15** z začetnim količnikom  $Q_0 = 0$ .
- 2d S primerjalnikom **6** testiraj, če je  $R_0 < D$ , in če je, potem končaj deljenje. Če je testiranje izvedeno v delilniku, testiraj, če je  $D = 0$ , in če je, postavi signal *deljenje-z-nič* ter končaj deljenje.
- 3d V vseh ostalih primerih uporabi poravnan delitelj  $A_{i+1}$  in vmesni količnik  $P_{i+1}$ , ki ju je general poravnalnik.
- 4d Izvedi odštevanje  $R_i - A_{i+1}$  s seštevalnikom **7** ter odštevanje  $R_i - \frac{1}{2}A_{i+1}$  s seštevalnikom **8**.
- 5d Uporabi izhodni signal prenosa PI iz seštevalnika **7** za nastavitev multiplekserjev **9** in **13**.

6d Shrani izhod multiplekserja **9** kot  $R_{i+1}$  v register za ostanek **5** ter shrani izhod iz OR-vrat **14** kot  $Q_{i+1}$  v register za količnik **15**.

7d S primerjalnikom **6** testiraj, če je  $R_{i+1} < D$ , in če je, končaj deljenje, sicer pa ponavlaj korake 3d do 7d.

### Izvedbeni primer

V tabeli 2 je prikazan celoten postopek deljenja z neprestanim poravnavanjem na primeru deljenja:  $197/9 = 21$  z ostankom 8, ki je izvedeno v 8-bitni binarni aritmetiki. Oznake so podobne kot v gornjem opisu z dodanima oznakama **R7** in **R8**, ki pomenita vhoda ali izhoda seštevalnikov **7** oziroma **8**. Delitelj  $D$  ostaja nespremenjen v času celotnega postopka deljenja in zato ni posebej prikazan.

Tabela 2: Primer računanja  $(11000101)_2 / (00001001)_2 = (00010101)_2$  z ostankom  $(00001000)_2$ , z uporabo deljenja z neprestanim poravnavanjem. Upoštevajmo, da sta  $A_{i+1}$  in  $P_{i+1}$  generirana s poravnalnikom, kot je bilo opisano v prejšnjem poglavju.

R7	R8	P	A	Q	opis	korak
11000101	11000101	.....	.....	00000000	naloži deljenec v R, resetiraj Q	1d,2d
		00010000	10010000		pripravi nova P in A	3d
<u>-10010000</u>	<u>-01001000</u>				odštej A in A/2	4d
(0)00110101	01111101				izhod <b>R7</b> je pozitiven, izberi P	5d
<b>00110101</b>		.....	.....	00010000	vpiši <b>R7</b> v R, aktiviraj poravnalnik	6d,7d
00110101	00110101	00000100	00100100		pripravi nova P in A	3d
<u>-00100100</u>	<u>-00010010</u>				odštej A in A/2	4d
(0)00010001	00100011				izhod <b>R7</b> je pozitiven, izberi P	5d
<b>00010001</b>		.....	.....	00010100	vpiši <b>R7</b> v R, aktiviraj poravnalnik	6d,7d
00010001	00010001	00000010	00010010		pripravi nova P in A	3d
<u>-00010010</u>	<u>-00001001</u>				odštej A in A/2	4d
(1)11111111	00001000				izhod <b>R7</b> je negativen, izberi P/2	5d
	<b>00001000</b>	.....	.....	000 <u>10101</u>	vpiši <b>R8</b> v R, $R < D \Rightarrow$ končano	6d,7d

### Zaključne opombe

Zaporedje predlaganih operacij delilnika z neprestanim poravnavanjem vodi do optimalne arhitekture podatkovno odvisnega delilnika. Posebej pomembna je razporeditev ničel in enic v binarni predstavitvi teh števil. Arhitektura delilnika je enostavna in kompaktna ter ne potrebuje dodatnih pomnilniških tabel.

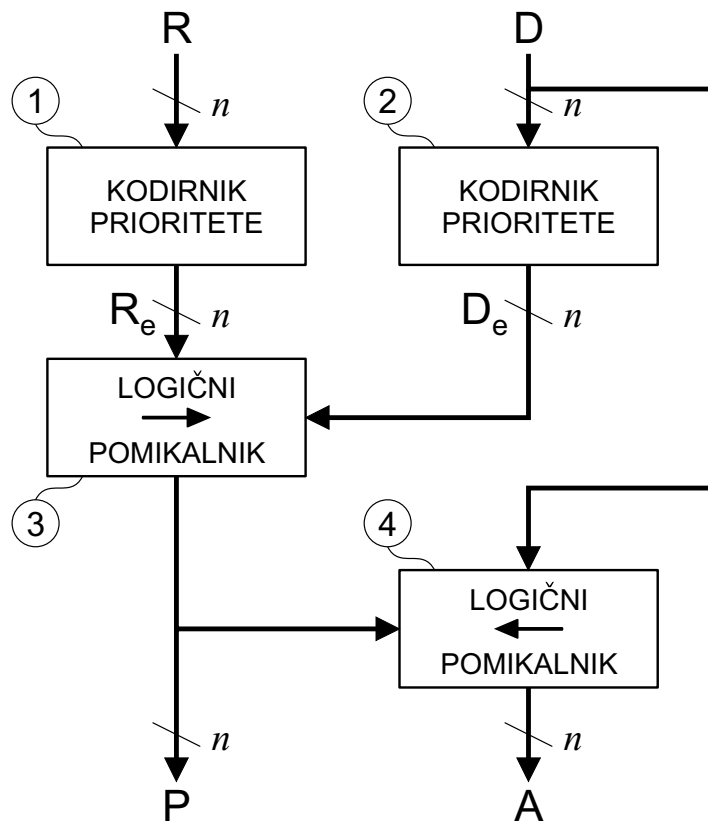
Celoten koncept delovanja hitrega delilnika z neprestanim poravnavanjem, ki ga predlagamo v tem dokumentu, je bil simuliran in izveden na programirljivih logičnih vezjih [9].

Ocena zmogljivosti in njena odvisnost od podatkov je bila podrobneje ocenjena na veliki množici naključno generiranih števil. Analiza rezultatov je potrdila, da je predlagan delilnik z vgrajenim poravnalnikom in z dvema seštevalnikoma boljši od vseh doslej poznanih konceptov podatkovno odvisnega deljenja.

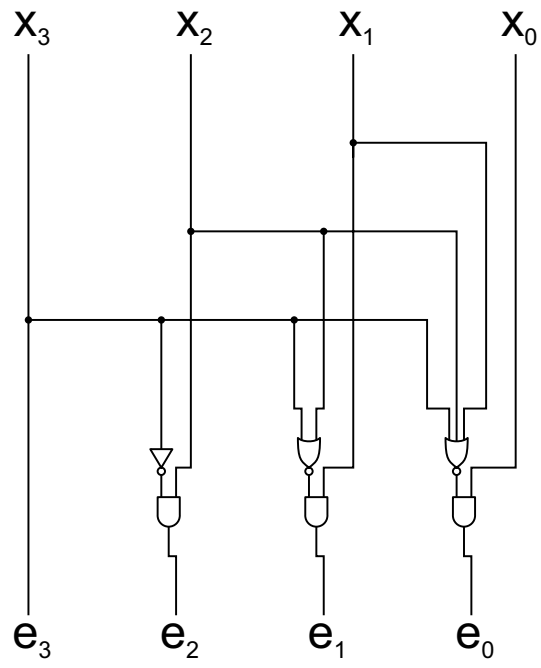
Predlagani postopek deljenja lahko izvedemo na kateremkoli številu bitov.

## Literatura

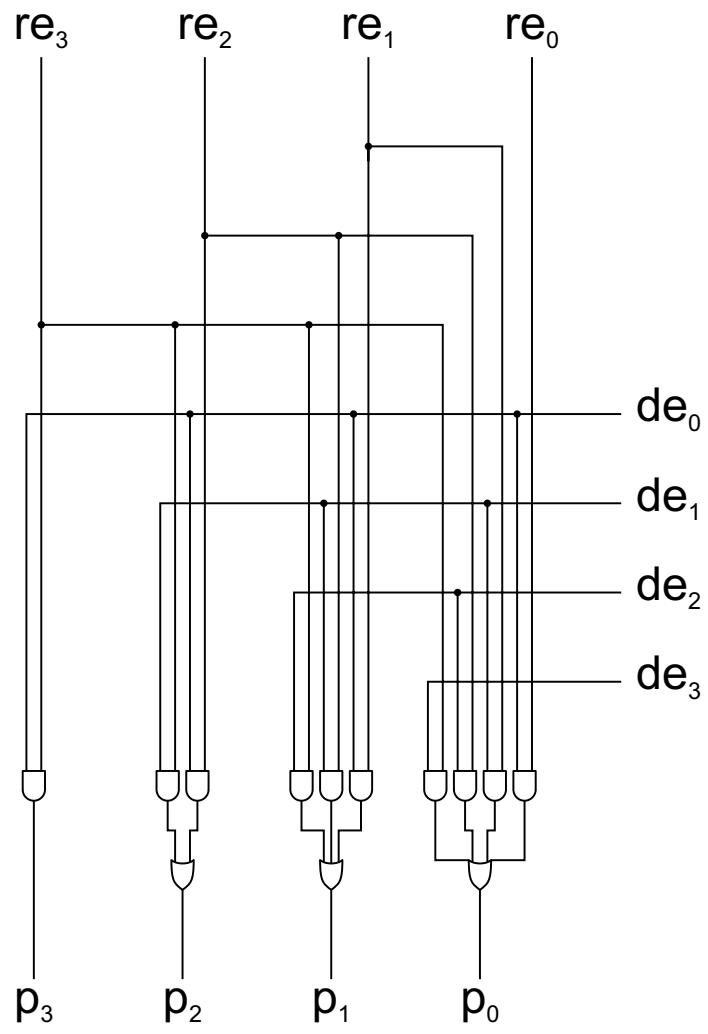
- [1] John L. Hennessy and David A. Patterson, *Computer Architecture—A Quantitative Approach*, 2nd ed., Morgan Kaufmann, San Francisco, California, 1996.
- [2] Israel Koren, *Computer Arithmetic Algorithms*, 2nd ed., A K Peters, Natick, Massachusetts, 2002.
- [3] Mi Lu, *Arithmetic and Logic in Computer Systems*, Wiley-Interscience, Hoboken, New Jersey, 2004.
- [4] Michael J. Flynn and Stuart F. Oberman, *Advanced Computer Arithmetic Design*, Wiley-Interscience, New York, New York, 2001.
- [5] David L. Harris, Stuart F. Oberman, and Mark A. Horowitz, *SRT Division Architectures and Implementations*, IEEE Symposium on Computer Arithmetic (1997).
- [6] Divider for Carrying Out High-Speed Arithmetic Operation, US-5014233, 1991.
- [7] Fast Arithmetic Modulo Divider, US-5493522, 1994.
- [8] A divider apparatus and associated method, EP-1351129A2, 2003.
- [9] Michael D. Ciletti, *Advanced Digital Design with the Verilog HDL*, Prentice Hall, Upper Saddle River, New Jersey, 2003.



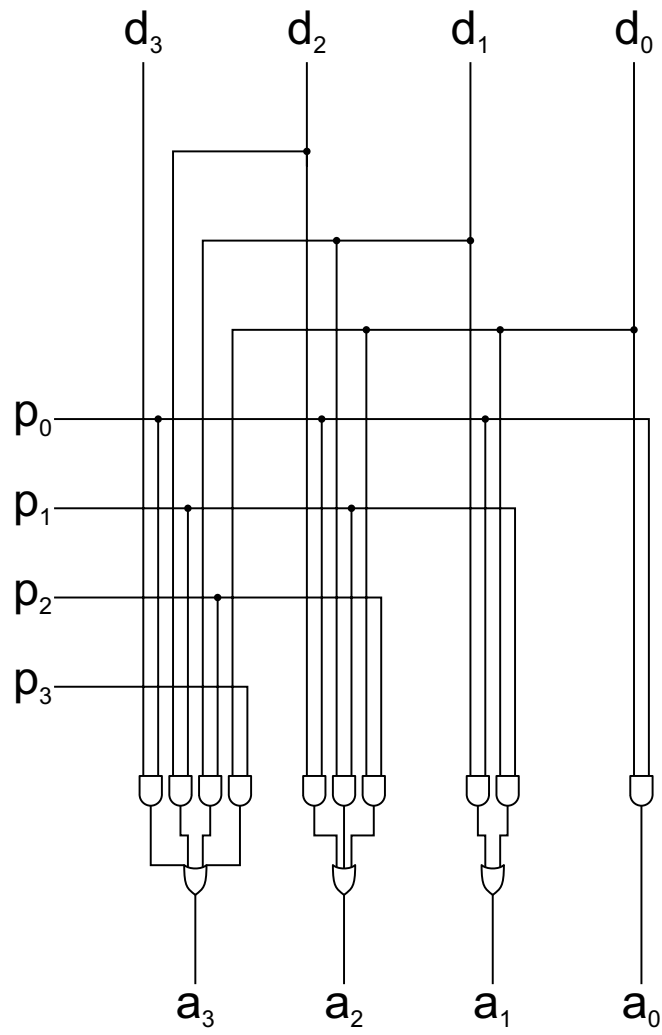
Slika 1: Blokovna shema poravnavalnika.



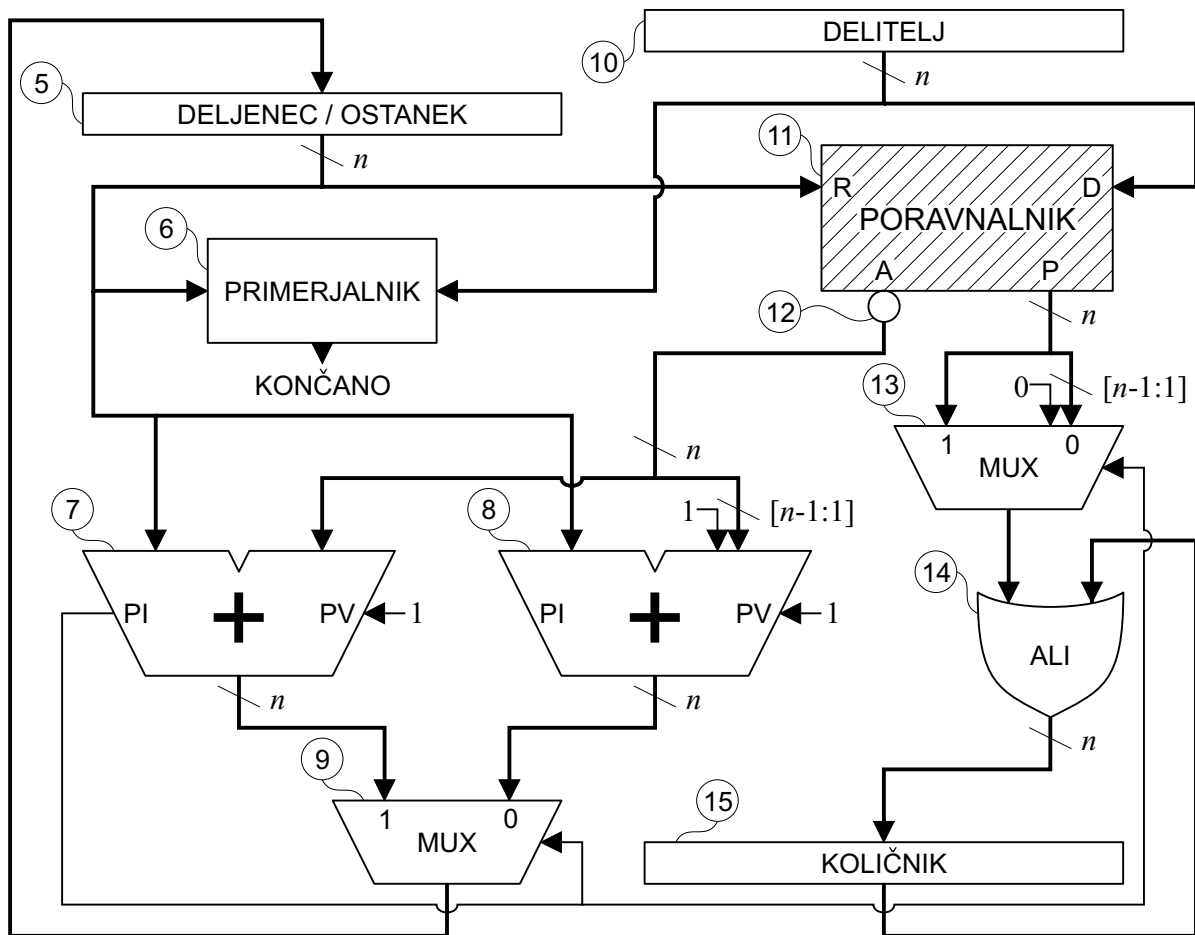
Slika 2: Blokovni diagram 4-bitnega kodirnika prioritete.



Slika 3: Blokveni diagram 4-bitnega logičnega pomikalnika v desno.



Slika 4: Blokovni diagram 4-bitnega logičnega pomikalnika v levo.



Slika 5: Blokovna shema aritmetičnega delilnika s poravnavanjem.

## Patentni zahtevki

1. Postopek za izvedbo deljenja z aritmetičnim delilnikom s poravnavanjem, označen s tem, da izvede aritmetično deljenje danega pozitivnega deljenca  $R_0$  z danim pozitivnim deliteljem  $D$  ter izračuna končni količnik  $Q_k$  in končni ostanek  $R_k$  z zaporedjem operacij:
  - 1d naloži register za ostanek z deljencem  $R_0$ , naloži register za delitelj z deliteljem  $D$  ter naloži register za količnik z začetnim količnikom  $Q_0 = 0$ ,
  - 2d s primerjalnikom testiraj, če je  $R_0 < D$ , in če je, potem končaj deljenje, če je izvedeno v delilniku, testiraj, če je  $D = 0$ , in če je, postavi signal *deljenje-z-nič* ter končaj deljenje,
  - 3d v vseh ostalih primerih uporabi poravnan delitelj  $A_{i+1}$  in vmesni količnik  $P_{i+1}$ , ki ju je generiral poravnalnik,
  - 4d hkrati izvede odštevanje  $R_i - A_{i+1}$  in  $R_i - \frac{1}{2}A_{i+1}$ ,
  - 5d uporabi izhodni signal prenosa PI iz seštevalnika za nastavitev obeh multiplekserjev,
  - 6d shrani izhod multiplekserja obeh izhodov seštevalnikov kot  $R_{i+1}$  v register za ostanek ter izhod iz OR-vrat kot  $Q_{i+1}$  v register za količnik,
  - 7d s primerjalnikom testiraj, če je  $R_{i+1} < D$ , in če je, končaj deljenje, sicer pa ponavljaj korake 3d do 7d.
2. Izum po zahtevku **1**, označen s tem, da je korak 3d, ki iterativno izvaja poravnavanje delitelja  $D$  in ostanka  $R$  ter generira vmesna poravnana količnik  $P$  in delitelj  $A$ , izveden v korakih:
  - 1p kodiraj prioriteto vmesnega ostanka  $R_i$  in delitelja  $D$  s kodirnikoma prioritete, ki generirata signala  $R_e$  in  $D_e$ , ki vsebujeta le vodilni, od nič različen bit v  $R_i$  oziroma  $D$ , kateremu sledijo same ničle,
  - 2p pomakni  $R_e$  za  $j$  bitov v desno, pri čemer je  $j$  število zadnjih ničel v  $D_e$ , z uporabo kontrolnega signala  $D_e$  v logičnem pomikalniku v desno, ki generira vmesni količnik  $P_{i+1}$ ,
  - 3p pomakni  $D$  za  $m - j$  bitov v levo, pri čemer je  $m - j$  število zadnjih ničel v vmesnem količniku  $P_{i+1}$ , z uporabo kontrolnega signala  $P_{i+1}$  v logičnem pomikalniku v levo, ki generira poravnan delitelj  $A_{i+1}$ .
3. Izum po zahtevkih **1** in **2**, označen s tem, da je relativni odmik med vodilnima, od nič različnima bitoma v  $R_i$  in  $D$  določen s pomikanjem vrednosti  $R_e$  v desno za toliko mest, kolikor je zadnjih ničel v  $D_e$ , in pomikanjem vrednosti  $D$  v levo, za toliko mest, kolikor je zadnjih ničel v že premaknjenem  $R_e$ , pri čemer je končni rezultat poravnave vmesni količnik  $P_{i+1}$  in poravnan delitelj  $A_{i+1}$ .
4. Izum po zahtevku **3**, označen s tem, da je pomikanje v desno in levo izvedeno z logičnima pomikalnikoma, sestavljenima iz  $n$  internih multiplekserjev, ki s pomočjo  $n$ -bitnega signala *ena-1* kombinatorično pomaknejo izhodno vrednost za zahtevano število mest, pri čemer je  $n$ -bitni *ena-1* signal enak signalu  $D_e$  za pomikanje v desno in signalu  $P_{i+1}$  za pomikanje v levo.

5. Izum po zahtevku **1**, označen s tem, da je korak odštevanja 4d izveden hkrati z dvema seštevalnikoma, od katerih eden izvede odštevanje  $R_i - A_{i+1}$  in drugi odštevanje  $R_i - \frac{1}{2}A_{i+1}$ , pri čemer da vsaj drugo odštevanje rezultat, ki je vedno pozitivno število.
6. Izum po kateremkoli prejšnjem zahtevku, označen s tem, da je zaznavanje izrednega primera  $D = 0$  izvedeno z znanim vezjem, kot na primer z OR-vrati, bodisi v delilniku ali izven njega.
7. Izum po kateremkoli prejšnjem zahtevku, označen s tem, da je kontrolno vezje za generiranje signalov *clock* in *write*, potrebnih za vpisovanje v register za deljenec/ostanek in register za količnik, izvedeno z znanim vezjem, kot na primer končnim avtomatom, bodisi v delilniku ali izven njega.

# ARITMETIČNI DELILNIK S PORAVNAVANJEM

## Povzetek

V patentni prijavi je opisana metoda za aritmetično deljenje pozitivnih celih števil s pripadajočim digitalnim vezjem delilnika, ki deli dani deljenec  $R_0$  z danim deliteljem  $D$  tako, da z zaporedjem operacij izračuna končni kvocient  $Q_k$  ter ostanek  $R_k$ . S predlagano metodo bo dosežen krajši čas deljenja za posamezne vrednosti obeh operandov. Pomembna inovativna dela delilnika sta *poravnalnik*, ki iz dveh vhodov, delitelja  $D$  in vmesnega ostanka  $R_i$ , naredi na izhodu poravnan delitelj  $A_{i+1}$ , in pripadajoči vmesni količnik  $P_{i+1}$ , ter dva *vzporedna* seštevalnika, ki zagotavljata hitrejšo izvajanje postopka deljenja. Aritmetični delilnik s poravnavanjem je izveden z naslednjim zaporedjem operacij: iz vhodov poravnalnika  $R_i$  in  $D$  odčitaj  $A_{i+1}$  in  $P_{i+1}$ ; odštej vzporedno  $R_i - A_{i+1}$  in  $R_i - \frac{1}{2}A_{i+1}$ ; če je rezultat  $R_i - A_{i+1}$  pozitiven, ga shrani kot vmesni ostanek  $R_{i+1}$  in  $Q_i + P_{i+1}$  kot vmesni količnik  $Q_{i+1}$ , sicer shrani rezultat  $R_i - \frac{1}{2}A_{i+1}$  kot vmesni ostanek  $R_{i+1}$  in  $Q_i + \frac{1}{2}P_{i+1}$  kot vmesni količnik  $Q_{i+1}$ ; ponavljaj naštete korake, dokler ne postane  $R_{i+1} < D$ .