IMPLEMENTACIJA PREDIKTORA ZA UPRAVLJANJE PRISTUPIMA SDRAM MEMORIJI
IMPLEMENTATION OF PREDICTOR FOR ACCESS CONTROLL TO SDRAM MEMORY

Nebojša Milenković, Vladimir Stanković, *Elektronski fakultet u Nišu*

Sadržaj – *Pristupi dinamičkoj (SDRAM) memoriji uključuju tri aktivnosti: pretpunjenje, aktiviranje (otvaranje) vrste i pristup koloni, svaka u trajanju petnaestak ili više nsec. Pri nizu sukcesivnih obraćanja kolonama u istoj vrsti banke, pretpunjenje i otvaranje vrste prethodi samo prvom obraćanju u tom nizu obraćanja, dok ostala obraćanja zahtevaju samo pristupe koloni. Jedan od zadataka kontrolera SDRAM memorija je da minimizira latenciju ili maksimizira propusnost pri pristupima memoriji. U prethodnim radovima autora predloženo je korišćenje hardverskih prediktora za minimiziranje latencije pri pristupima SDRAM memoriji. Prediktori dinamički prate istoriju obraćanja SDRAM memoriji, i na osnovu toga predviđaju kada treba zatvoriti otvorenu vrstu (čime se isključuje vreme pretpunjenja) i koju vrstu treba otvoriti kao sledeću aktivnu vrstu (čime se isključuje vreme otvaranja vrste) pre sledećeg pristupa u okviru iste banke. U ovom radu izloženo je projektovanje na nivou funkcionalnih blokova hadvera prediktora zatvorene vrste i prediktora otvorene vrste kao elemenata kontrolera SDRAM memorija.*

Ključne reči: SDRAM, kontroler SDRAM-a, latencija, prediktor zatvorene vrste, prediktor otvorene vrste.

Abstract – *Dynamic memory accesses include three activities: precharge, row activation and column access, each of about 15 ns or more. For a sequence of successive accesses to columns into the same row of a bank, precharge and row activation must precede only the first access in the sequence, and subsequent accesses require only column accesses. One task of SDRAM controller is to minimize latency or to maximize bandwidth during memory accesses. Authors have proposed usage of hardware predictors to minimize latency of SDRAM memory accesses in their previous papers. Predictors dynamically register SDRAM access history, and based on them predict when to close opened row (this excludes precharge time) and which next row to open (this excludes row activation time) before next access into the same bank. In this paper design of hardware for closed row predictor and opened row predictor as elements of SDRAM memory controller is explained on functional blocks level.*

Key words: SDRAM, SDRAM controller, latency, closed row predictor, opened row predictor.

1. UVOD

Već duže od trideset godina glavne memorije računara najčešće se implementiraju dinamičkim RAM (DRAM) memorijama. Savremeni tipovi ovih memorija sa najraširenijom primenom su sinhrone DDR2 i DDR3 SDRAM memorije. Čipovi ovakvih memorija sadrže po 4 ili 8 nezavisnih banaka memorije, sa pratećim kolima za pristup lokacijama u njima i privremeno čuvanje pročitanih podataka u pridruženim sens pojačavačima. U tipičnim aplikacijama DRAM memorije se u računaru nalaze iza keš memorija sa kojima razmenjuju blokove podataka dužine 32 bajta (B) do 128B. Punjenje keš memorija promašenim blokovima podataka značajno utiče na performanse računara kao celine. Vreme za punjenje keša promašenim blokovima podataka T_M uključuje latenciju DRAM memorije T_A i vreme prenosa bloka podataka T_T, $T_M = T_A + T_T$. Fizika rada i implementacija DRAM memorija zahtevaju da se pri pristupu obave tri aktivnosti: pretpunjenje, aktiviranje (otvaranje) vrste i pristup koloni, koje se izvršavaju u vremenskim intervalima t_{RP} , t_{RA} i t_{CL} respektivno. Latencija memorije T_A jednaka je zbiru ova tri vremenska intervala $T_A = t_{RP} + t_{RA} + t_{CL}$. Svi podaci pročitani iz jedne vrste prilikom njenog aktiviranja prisutni su u sens pojačavačima (kao baferu vrste) sve do zatvaranja te vrste sledećim pretpunjenjem. Ovo omogućuje da se po otvaranju vrste sledeća uzastopna obraćanja kolonama u toj vrsti svedu samo na pristupe kolonama, sa vremenom $T_A = t_{CL}$. Postoji mogućnost da se pristupima pri čitanju ili upisu pridoda i pretpunjenje, za čime onda nema potrebe pri sledećem obraćanju nekoj drugoj vrsti u banci, pa se latencija onda svodi na $T_A = t_{RA} + t_{CL}$. Ovo pokazuje da DRAM memorije nisu u strogom smislu memorije sa slučajnim pristupom, koje karakteriše vreme pristupa nezavisno od adresa lokacija kojima se pristupa, već su memorije sa bankama, vrstama i kolonama kao dimenzijama koje određuju vreme pristupa. Pošto vreme pristupa T_A može imati tri veličine: $t_{RP}+t_{RA}+t_{CL}$, $t_{RA}+ t_{CL}$ ili samo t_{CL} , istražuju se rešenja koja u pojedinim oblastima aplikacija mogu smanjivati latenciju pristupa ka donjoj granici t_{CL} [1,2]. Aktuelne DDR3 i DDR2 SDRAM memorije imaju vremena t_{RP}, t_{RA} i t_{CL} veličina 12÷18 ns. Ove memorije rade u režimu prenosa grupe podataka (burst mode) dužine 8 ili 4 reči, i pri brzini prenosa 1333MTps, prenos bloka podataka dužine 64 bajta preko 64-bitne magistrale podataka traje samo T_T = 6ns. Ovo pokazuje da u vremenu T_M latencija DRAM memorija ima dominantan uticaj.

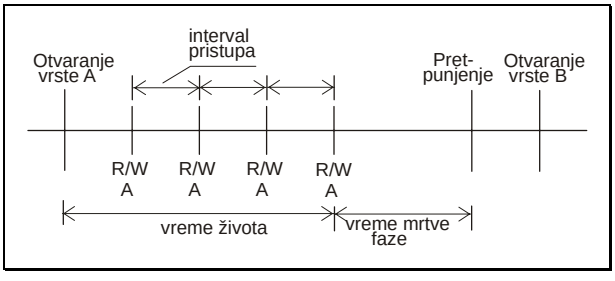
Pogoršanje performansi procesora zbog čekanja na pribavljanje podataka iz glavne memorije smanjuje se korišćenjem neblokirajućih keš memorija, pribavljanjem podaka unapred i drugim rešenjima [3,4,5], koja imaju svoju cenu i ograničenja. Kao korisnu dopunu ovih rešenja autori su predložili i istražili korišćenje prediktora u upravljanju pristupima DRAM memoriji [6,8], koji mogu u značajnom broju slučajeva unapred naložiti pretpunjenje i otvaranje nove vrste, i time eliministi vremena t_{RP} i t_{RA} pri novim pristupima.

U daljem tekstu u odeljku 2 ukratko su izložene osnove rada dva prediktora koji unapred zatvaraju otvorenu vrstu, zaključujući na osnovu predistorije da ta vrsta dalje neće biti aktivna. Na osnovu toga predložene su hardverske implementacije ovih prediktora. Odeljak 3 sadrži prikaz osnova rada prediktora koji unapred otvara novu vrstu, u koju će verovatno biti usmeren sledeći pristup u okviru banke, i hardverske implementacije ovog prediktora. Hardverske implementacije prikazane su na nivou funkcionalnih blokova, što omogućuje približnu procenu složenosti ovih prediktora. Odeljak 4 sadrži zaključak, a odeljak 5 literaturu.

U radu su prikazane implementacije prediktora za DRAM memoriju organizacije 2 grupe čipova sa po 64 linija podataka, ukupnog kapaciteta 2GB. U svakoj grupi imamo po 8 DDR3 SDRAM čipova kapaciteta 1Gbit, sa 8 banaka unutar čipa, svaka sa 8K vrsta sa 2K 8-bitnih kolona u svakoj vrsti.

2. PREDIKTORI ZATVORENE VRSTE

Rad prediktora zatvorene vrste zasniva se na praćenju vremenskih intervala između sukcesivnih pristupa u okviru iste banke DRAM-a. Sledeća vremena karakterišu takve pristupe. Vreme života je vreme koje protekne od prvog pristupa u neku vrstu banke DRAM memorije do poslednjeg pristupa u tu vrstu, pre njenog zatvaranja pretpunjenjem. Vreme mrtve faze je vreme koje protekne od poslednjeg pristupa u neku vrstu DRAM memorije do njenog zatvaranja. Interval pristupa je vreme koje protekne između dva sukcesivna pristupa istoj vrsti DRAM memorije. Ovo je prikazano na slici 1. Ukoliko nakon otvaranja neke vrste u nju više nemamo nijedan pristup pre njenog zatvaranja, tada je taj prvi pristup ujedno i poslednji, pa je tada vreme života te vrste jednako nuli, to jest imamo pojavu nultog vremena života.



Sl. 1. Karakteristična vremena pri pristupima DRAM-u

Simulacijom ponašanja SDRAM memorija pri izvršenju grupe benchmark programa iz SPECint95 ustanovili smo sledeće odnose između intervala pristupa i vremena mrtve faze. Ako nakon poslednjeg pristupa otvorenoj vrsti protekne vreme nekoliko puta duže od poslednjeg intervala pristupa, velika je verovatnoća da je vrsta ušla u mrtvu fazu, pa je treba zatvoriti komandom pretpunjenja. Već dvostruko duže vreme od poslednjeg intervala pristupa daje dobru indikaciju ulaska vrste u mrtvu fazu. Upravo ovo iskoristili smo da definišemo rad prediktora mrtve faze i njegovu hardversku implementaciju.

Simulacijom smo takođe ustanovili da je vreme života znatnog broja tek otvorenih vrsta jednako nuli. Ovo upućuje da vrste sa nultim vremenom života odmah posle otvaranja i čitanja ili upisa treba zatvoriti pridodavanjem i komande pretpunjenja. Dinamičkim praćenjem istorije tipa nulto-nenulto vreme života svake novo otvorene vrste može se sa znatnom izvesnošću predvideti da li će pri ponovnom otvaranju vrsta imati nulto vreme života.

2.1 *Prediktor nultog vremena života*

Za beleženje istorije nulto-nenulto vreme života koristimo dvobitne brojače sa zasićenjem, sa stanjima {0,1,2,3} i početnim stanjem 0. Neka stanja 0 i 1 ukazuju da pri sledećem otvaranju vrsta neće imati nulto vreme života, a stanja 2 i 3 da će imati nulto vreme života. Nulto vreme života po otvaranju vrste zahteva inkrementiranje stanja brojača, osim ako je ono već bilo 3, a nenulto vreme života zahteva dekrementiranje stanja brojača, osim ako je ono već bilo 0. Rezultati simulacije pokazali su da se neznatno lošiji rezultati predviđanja dobijaju ako se po jedan ovakav brojač koristi za grupu od 16 sukcesivnih vrsta, što broj potrebnih brojača za razmatranu konfiguraciju DRAM memorija smanjuje sa 128K na samo 8K. Na osnovu tih zaključaka razradili smo prediktor nultog vremena života.

Kontroler DRAM-a ima dva reda za prihvatanje naloga:
- red sa tekućim nalozima pristiglim od procesora-keša ili DMA kontrolera (RED_A),
- red sa nalozima za pretpunjenje iz prediktora mrtve faze (RED_B).

Opsluživanje naloga iz reda RED_A ima prioritet u odnosu na opsluživanje naloga iz reda RED_B.

U kontroleru DRAM memorije za svaku banku B_k, k=1,2,…,N, uveden je:

- Po jedan Registar Adrese Otvorene Vrste ($RAOV_k$) u kome se čuva adresa trenutno otvorene vrste u banci B_k. Neka je R_i =($RAOV_k$). Označimo sa R_j adresu vrste kojoj se trenutno pristupa u banci B_k, na osnovu naloga iz reda RED_A.
- Po jedan Registar Adrese Zatvorene Vrste ($RAZV_k$) u kome se čuva adresa vrste zatvorene neposredno pre ili pri otvaranju tekuće otvorene vrste u banci B_k.
- Po jedan indikator Ponovnog Pristupa Vrsti (PPV) sa početnim stanjem 0

Korišćenje ovakvih prediktora nultog vremena života organizujmo uz pomoć sledeće dve funkcije.

Čitanje predikcije:

Pri aktiviranju sledeće vrste R_j u banci B_k čita se stanje prediktora pridruženog toj vrsti.

A1. Ako je stanje prediktora $\in\{2,3\}$ (prediktor svojim stanjem prognozira nulto vreme života aktivirane vrste), primljenom nalogu od procesora (čitanje, upis) kontroler DRAM-a pridodaje i komandu pretpunjenja (čitanje sa pretpunjenjem, upis sa pretpunjenjem). Preći na A3.

A2. Ako je stanje prediktora $\in\{0,1\}$ (prediktor svojim stanjem prognozira nenulto vreme života aktivirane vrste), primljeni nalog od procesora (čitanje, upis) kontroler DRAM-a prosleđuje u vidu komande (čitanje, upis).

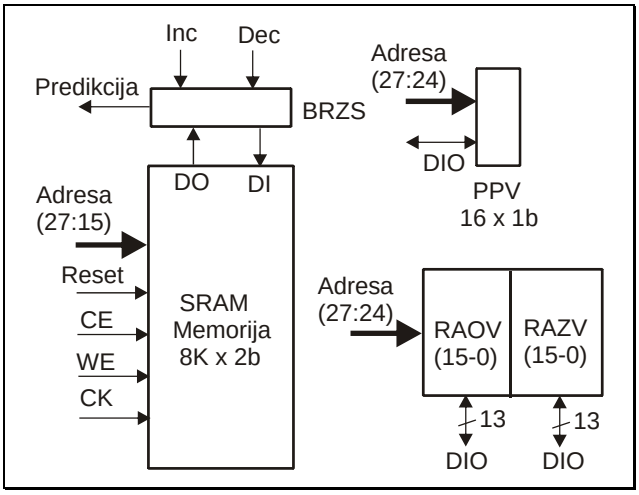
A3. Kraj. Preći na ažuriranje prediktora nultog vremena života.

Ažuriranje prediktora:

B1. Ako je pristup u banci B_k usmeren u pogrešno zatvorenu vrstu $R_j = R_i$ (ima se nenulto vreme života) i PPV_k=0, postaviti $PPV_k \leftarrow 1$ i dekrementirati sa zasićenjem 2-bitni prediktor. Ako je PPV_k već jednako 1 ne činiti ništa. Preneti upravljanje na prediktor mrtve faze.

B2. Ako je pristup u banci B_k usmeren u novu vrstu $R_j \neq R_i$, a za banku je PPV_k=0 (bilo je nulto vreme života vrste R_i), prediktor za vrstu R_i inkrementirati sa zasićenjem. Vrsta R_i određena je sadržajem registra adrese otvorene vrste $RAOV_k$ u banci B_k. Tek posle ažuriranja prediktora za vrstu R_i i prenosa $RAZV_k \leftarrow (RAOV_k)$ u registar $RAOV_k$ može se upisati adresa nove aktivirane vrste R_j.

B3. Ako je pristup u banci B_k usmeren u novu vrstu $R_j \neq R_i$, a za banku je PPV_k=1 (bilo je nenulto vreme života vrste R_i), prediktor za vrstu R_i dekrementirati sa zasićenjem, a PPV_k obrisati ($PPV_k \leftarrow 0$), i u kontroleru DRAM-a za banku B_k upisati adresu R_j u registar $RAOV_k$.



Sl. 2 Elementi prediktora nultog vremena života

Elementi prediktora nultog vremena života prikazani su na slici 2. Svih 8K 2-bitnih prediktora locirano je u SRAM memoriji, adresiranoj sa 13 najznačajnijih od 28 bitova adresa podataka u SDRAM memoriji. Čitanje stanja izabranog prediktora kao i njegovo ažuriranje obavlja se u 2-bitnom brojaču sa zasićenjem BRZS. Indikatori PPV_k grupisani su u polje indikatora, a registri $RAOV_k$ i $RAZV_k$ u polja registara, k=1,2,...,N. Adresiranje indikatora PPV_k i registara $RAOV_k$ i $RAZV_k$ vrši se sa 4 bita adrese najveće težine.

2.2 *Prediktor mrtve faze*

Ako zatvaranje otvorene vrste ne naloži prediktor nultog vremena života, zaduženje za to prenosi se na prediktor mrtve faze. To se dešava i kada prediktor nultog vremena života pogrešno zatvori vrstu, koja pri sledećem obraćanju banci biva ponovo otvorena. Da prediktor mrtve faze zatvara vrste sa nenultim vremenom života, trebalo bi da on za svaku banku B_k, k=1,...,N, u SDRAM memoriji ima po:

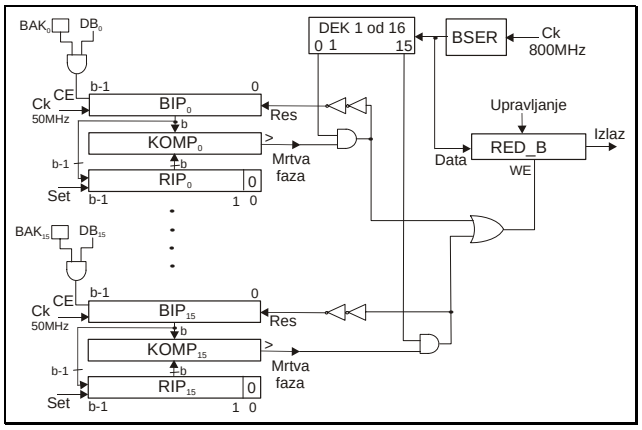
- Brojač Intervala Pristupa BIP_k formata BIP_k [b-1:0],
- Registar Intervala Pristupa RIP_k formata RIP_k [b-1:0], sa bitom RIP_k [0] fiksiranim na nuli,
- Komparator (b-to bitni) sadržaja brojača BIP_k i registra RIP_k,
- Indikator Banka Aktivna (BAK_k) koji stanjem 1 ukazuje da je u banci k neka vrsta aktivirana,
- Flip- flop Dozvole Brojanja (DB_k).

Brojači Intervala Pristupa pridruženi bankama okidaju se taktnim signalom učestanosti 50MHz. Pošto interfejs kontrolera DRAM-a sa DRAM-om radi na 800MHz, to se okidni signal za ove registre dobija deobom signala od 800MHz sa 16. Kako je taktni signal procesora 3,2GHz, ovih 50MHz odgovaraju deobi taktnog signala procesora sa 64. Ovo znači da jedinični inkrement Brojača Intervala Pristupa uključuje 64 jedinice intervala pristupa merenih periodama takta procesora. U postupku inicijalizacije kontrolera SDRAM-a svi registri RIP postavljaju se na maksimalnu vrednost.

C1. Pri aktiviranju nove vrste R_j u banci B_k brojač BIP_k se resetuje a indikator PPV_k se briše. Daje se dozvola da taktni signal od 50MHz okida brojač BIP_k ($DB_k \leftarrow 1$).

Posle svakog taktnog ciklusa signala od 50MHz kojim se okidaju brojači BIP za sve banke u kojima postoje otvorene vrste, komparatori pridruženi parovima (BIP, RIP) izlaznim signalom 1 pri $(BIP_k[b\text{-}1:0]) > (RIP_k[b\text{-}1:0])$ pokazuju na banke u kojima su otvorene vrste ušle u mrtvu fazu. Za one banke za koje je taj uslov ispunjen ukida se dozvola brojanja za njihove brojače BIP i njihovi identifikatori ID upisuju se u RED_B komandi pretpunjenja. Za slučaj da ima više takvih banaka mehanizam serijalizacije upravlja upisima tih identifikatora u ovaj red.

C2. Ako je sledeće obraćanje u banci B_k usmereno u otvorenu vrstu $R_j = R_i$ naložiti prenos $RIP_k[b\text{-}1:1] \leftarrow (BIP_k[b\text{-}2:0])$, zatim resetovati BIP_k i dozvoliti brojanje za ovaj brojač ($DB_k \leftarrow 1$).



Sl.3 Elementi prediktora mrtve faze

Elementi prediktora mrtve faze prikazani su na slici 3. Rezultati simulacije pokazali su da je za praćenje intervala pristupa na korišćenoj grupi benchmark programa dovoljno korišćenje brojača BIP i registara RIP dužina b=10 binarnih pozicija. BSER je 4-bitni brojač koji generiše adrese banaka i

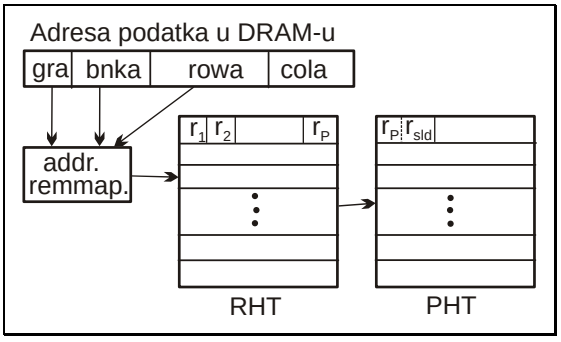
koji zajedno sa dekoderom tipa 1 od 16 obavlja funkciju serijalizacije upisa u RED_B identifikatora ID banaka čije otvorene vrste su ušle u mrtvu fazu. BSER se okida taktnim signalom učestanosti 800MHz. Po zapisivanju identifikatora banke za koju je ispunjen kriterijum ulaska u mrtvu fazu, njen brojač BIP automatski se resetuje. Upravljačka jedinica kontrolera DRAM memorije upravlja pristupima registrima u okviru reda RED_B, prenosima $RIP_k[b\text{-}1:1] \leftarrow (BIP_k[b\text{-}2:0])$, stanjima signala BAK i DB i drugim aktivnostima prediktora mrtve faze.

U radu [7] prikazan je prediktor koji unapred zatvara otvorenu vrstu organizovan po analogiji sa prediktorima grananja u dva nivoa.

3. PREDIKTOR SLEDEĆE VRSTE

Blagovremeno zatvaranje otvorene vrste eliminiše potrebu za pretpunjenjem pri sledećem pristupu nekoj drugoj vrsti u banci. Ostaje potreba za otvaranjem te vrste pre pristupa podacima u odgovarajućim kolonama. Dinamičkim praćenjem redosleda otvaranja vrsta u banci, posle zatvaranja neke vrste moguće je predvideti kojoj vrsti će biti upućen sledeći pristup, pa unapred otvoriti tu vrstu. Time se u vremenu pristupa T_A pored vremena t_{RP} eliminiše i vreme t_{RA}, svodeći vreme pristupa na teorijski minimum t_{CL}. Poslednjih godina mnogi autori predlagali su rešenja za efikasno pribavljanje unapred blokova podataka u keš memoriju. Rešenja izložena u radovima [3,4] inspirisala su nas da slično rešenje predložimo za organizovanje prediktora sledeće vrste. Zasnovanost ovog rešenja sa rezultatima simulacije koji potvrđuju dobru tačnost predviđanja koju vrstu otvoriti unapred izložena je u radu autora [8].

Funkcionalna organizacija prediktora koji predviđa koju sledeću vrstu otvoriti data je na slici 4. Prediktor se sastoji od dve tabele: Tabele istorije vrsta (Row History Table-RHT) i



Sl. 4. Organizacija prediktora koji predviđa sledeću vrstu

Tabele istorije obrazaca (Pattern History Table - PHT). RHT pamti poslednjih p vrsta kojima se pristupalo za svaku od banaka, što znači da RHT ima onoliko stavki koliko ima banaka. PHT sadrži predikcije, i ima m stavki, pri čemu je maksimalna vrednost za m broj vrsta u banci. Svaka stavka sadrži j dvodelnih polja: trenutnu vrstu (r_p) i sledeću vrstu (r_{sld}). Indeks za pristup PHT dobija se kao t najnižih bitova sabiranja sa odsecanjem poslednjih p indeksa vrste iz stavke za tu banku u RHT, tako da je $m=2^t$.

Prediktor koji predviđa koju sledeću vrstu treba otvoriti realizuje se preko dve osnovne funkcije Ažuriranje i Čitanje predikcije. Ako sa *tekuća_banka* i *nova_vrsta* označimo banku u kojoj se javio pristup i adekvatnu novu vrstu, tada bi ove funkcije izgledale ovako.

Ažuriranje prediktora:

Ova operacija vrši ažuriranje RHT i PHT kod otvaranja nove vrste, tako da istorija zapamćena u ovim tabelama bude validna i sastoji se iz sledećih koraka:

D1. Za početak, *tekuća_banka* se koristi kao indeks za pristup RHT, i locira se sekvenca vrsta ($vrsta_1$, $vrsta_2$, ... $vrsta_p$). Ova sekvenca koristi se za indeksiranje PHT i locira se odgovarajuća stavka u njoj.

D2. Sekvenca vrsta ($vrsta_1$, $vrsta_2$, ... $vrsta_p$) zamenjuje se sekvencom ($vrsta_2$, ... $vrsta_p$, *nova_vrsta*).

D3. Od svih dvodelnih polja u izabranoj stavci PHT, locira se polje koje počinje sa $vrsta_p$. Ukoliko takvo polje postoji, deo koji predviđa sledeću vrstu u lociranom polju zamenjuje se sa *nova_vrsta*. Kraj operacije Ažuriranje.

D4. Ukoliko takvo polje ne postoji, proverava se da li su sva dvodelna polja u datoj stavci popunjena. Ako jesu, vrši se izbor polja koje će biti izbačeno, koristeći FIFO algoritam, umesto kojeg će biti ubačeno novo polje.

D5. Ubacuje se novo dvodelno polje. U njemu se kao prvi deo polja upisuje $vrsta_p$ a kao drugi deo upisuje *nova_vrsta*. Kraj operacije Ažuriranje.

Korakom D3, odnosno D5 *nova_vrsta* se definiše kao sledeća vrsta koja neposredno sledi za sekvencom ($vrsta_1$, $vrsta_2$, ..., $vrsta_p$).

Čitanje predikcije:

Ova operacija predviđa sledeću vrstu, na bazi tekuće, i na bazi informacije da je prethodna sekvenca vrsta kojima se pristupalo u datoj banci ($vrsta_1$, $vrsta_2$, ... $vrsta_p$).

E1. Oznaka banke *tekuća_banka* se koristi kao indeks za pristup RHT, i locira se adekvatna sekvenca vrsta ($vrsta_1$, $vrsta_2$, ... $vrsta_p$).

E2. Sekvenca vrsta ($vrsta_1$, $vrsta_2$, ... $vrsta_p$) koristi se za izračunavanje indeksa za lociranje stavke u PHT.

E3. Proverava se da li neko od polja u datoj stavki u svom prvom delu sadrži $vrsta_p$. Ukoliko takvo polje ne postoji, tada za taj slučaj ne postoji predikcija i to je kraj operacije Čitanje predikcije. Ukoliko takvo polje postoji, nastavlja se sa sledećim korakom.

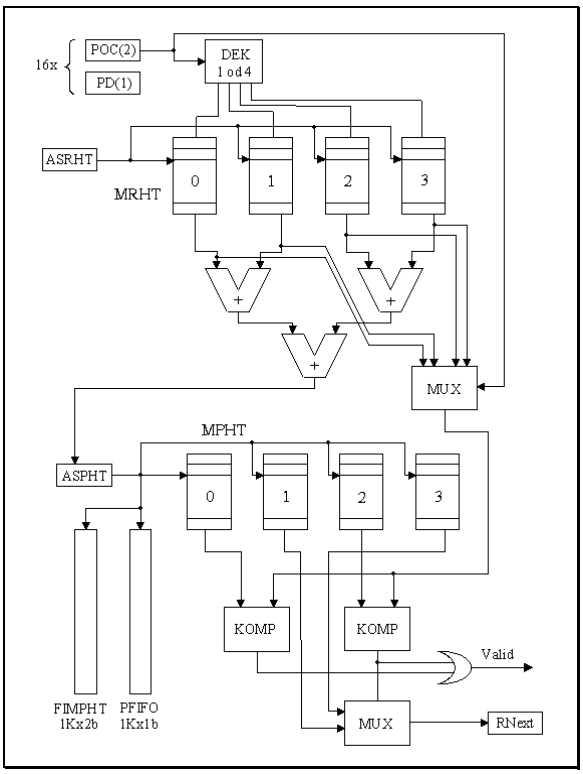
E4. Selektuje se drugi deo nađenog polja, koji predviđa sledeću vrstu (r_{sld}), pošto on predviđa sledeću vrstu kao nastavak sekvence ($vrsta_1$, $vrsta_2$, ... $vrsta_p$).

E5. Konačno, kontroler DRAM memorije koristi r_{sld} kako bi inicirao otvaranje te vrste u datoj banci.

Implementacija prediktora koji predviđa sledeću vrstu zahteva $g \cdot b \cdot p \cdot \lceil \log_2 n \rceil$ bitova za RHT i $m \cdot j \cdot 2 \cdot \lceil \log_2 n \rceil$ bitova za PHT. Ovde je g broj grupa DRAM čipova kojima se pristupa istovremeno, b broj banaka u svakoj grupi čipova, a n je broj vrsta u jednoj banci. Takođe, potreban je jedan t-bitni sabirač i dodatna logika za implementaciju operacija Ažuriranje i Čitanje predikcije. Za usvojenu DRAM strukturu od 2GB, korišćene su sledeće vrednosti: p=4, j=2, i dve vrednosti za m: 4096 i 1024. Za izabrane vrednosti, potrebno je 832b za RHT i 26KB za PHT, ako je m=4096, odnosno 6.5KB za PHT, ako je m=1024.

Šematski prikaz implementacije prediktora sledeće vrste prikazan je na slici 5. RHT tabelu sa 2×8 = 16 stavki implementirajmo u memoriji MRHT organizovanoj kao 4 paralelne memorije MRHT(0), …, MRHT(3) tipa 16×13 bita. Ovde je jedna stavka RHT tabele sa adresama poslednje 4 otvorene vrste u jednoj banci smeštena na istim relativnim

adresama u okviru ove 4 paralelne memorije, čime je omogućeno istovremeno čitanje sve 4 vrste iz date stavke RHT tabele. Na taj način one se mogu sabrati uz pomoć 3 sabirača ne zahtevajući predugo vreme sabiranja. Obzirom da se ovim sabiranjem formira adresa za pristup PHT tabeli, koja ima 1K stavki, sabirači su 10-bitni (koriste se niža 10 bita 13-bitnih adresa vrsta iz odgovarajuće stavke RHT-a), sa odbacivanjem prenosa. ASRHT je 4-bitni registar koji u memorijama MRHT(0), …,MRHT(3) adresira lokacije koje sadrže elemente iste stavke RHT tabele. Za ažuriranje stavki RHT-a, lokacije sa istim adresama u MRHT(0), …, MRHT(3) organizujmo u obliku kružnog reda, sa 2-bitnim pokazivačem na početak reda POC, u koji se upisuje adresa poslednje otvorene vrste. Svakoj stavci RHT-a pridružimo po indikator PD (Predikcija Dopuštena), sa početnim stanjem 0, koji se postavlja kada se u odgovarajuću stavku RHT-a upišu 4 adrese otvorenih vrsta u odgovarajućoj banci. Ovih pokazivača i indikatora ima po 16, za svaku stavku RHT-a po jedan: POC[0], …, POC[15] i PD[0], …, PD[15]. Pokazivači POC implementirani su kao brojači sa upravljačkim ulazima za inkrementiranje INK i resetovanje RES. Indikatori PD brišu se resetom, a postavljaju signalom prenosa iz odgovarajućih brojača POC.



Sl. 5. Hardver prediktora sledeće otvorene vrste

Tabela PHT organizovana je kao 4 paralelne memorije MPHT(0), …, MPHT(3), svaka organizacije 1Kx13 bita, pri čemu je svaka stavka PHT tabele smeštena u 4 lokacije paralelnih memorija koje imaju istu relativnu adresu, čime je omogućeno istovremeno čitanje sve 4 lokacije. Ove 4 lokacije sadrže 2 para vrsta-sledeća_vrsta, odnosno r_p-r_{sld}. ASPHT je 10-bitni registar pokazivač stavke PHT-a, koji adresira lokacije MPHT(0), …, MPHT(3) sa istim adresama. Prilikom pristupa PHT-u koriste se i 2 bita koji određuju kojoj se od 4 paralelne memorije pristupa. Ova 2 bita određuju se uz pomoć FIMPHT i PFIFO. FIMPHT, koji je organizacije 1Kx2 bita, za svaku stavku PHT-a sadrži po dva bita koja ukazuju na popunjenost prvog, odnosno drugog para r_p-r_{sld}, čime se definiše upis na odgovarajuće mesto. U slučaju da su oba para popunjena PFIFO, koji je organizacije 1Kx1b definiše u koji od dva para će se obaviti upis, odnosno koji par će biti izbačen. Tom prilikom se odgovarajući bit PFIFO komplementira. U fazi inicijalizacije se sve pomoćne strukture resetuju (brišu): POC[0], …, POC[15], PD[0], …, PD[15], kompletan FIMPHT i kompletan PFIFO.

4. ZAKLJUČAK

U radu je izložena implementacija prediktora koji u DRAM memoriji unapred zatvaraju otvorenu vrstu i unapred otvaraju vrstu u koju će verovatno biti upućen sledeći pristup. Organizacija hardvera tri tipa prediktora prikazana je na nivou funkcionalnih blokova. Ovakva organizacija hardvera prediktora, zajedno sa blokom upravljanja koji organizuje rad ovih prediktora na opisani način, sintetizovana je na FPGA čipu familije Xilinx SpartanII, model xc2v500-6fg256. Skraćenje prosečne latencije DRAM memorija koje se može postići primenom ovih prediktora, navedeno u radu [8], opravdava povećanje cene realizacije kontrolera DRAM memorije sa pridodatim ovde prikazanim prediktorima.

5. LITERATURA

[1] Rixner, S."Memory controller optimization for Web servers", *Proc. MICRO-37*, 2004., pp. 355-366.

[2] V. Stankovic, N. Milenkovic, "Access Latency Reduction in Contemporary DRAM Memories", *Facta Universitatis, series: Electronics and Energetics,* Vol. 17, No. 1, April 2004, pp. 81-97.

[3] Z. Hu, S. Kaxiras, M. Martonosi, "Timekeeping in the Memory System: Predicting and Optimizing Memory Behavior", *The 2003 IEEE International Solid_state Circuits Conference (ISSCC 2003)*, February 2003.

[4] A. Lai, C. Fide, B. Falsafi, "Dead-Block Prediction and Dead-Block Correlating Prefetchers", *Proc. 28th ISCA*, June 2001, pp. 144-154.

[5] Hu,Z., M.Martonosi and S.Kaxiras,"TCP: Tag Correlating prefetchers", *Proc. 9th HPCA*, 2003, pp.317-326.

[6] V. Stankovic, N. Milenkovic, "DRAM Controller with a Close-Page Predictor", *Eurocon 2005 - The International Conference on "Computer as a Tool*, Belgrade, November 2005, pp. 693-696.

[7] Xu,Y., A.S.Agarwal, and B.T.Davis,"Prediction in dynamic SDRAM controller policies", in *SAMOS, LNCS 5657*, pp. 128-138, Springer 2009.

[8] V. Stankovic, N. Milenkovic, "DRAM Controller with a Complete Predictor", *IEICE Transactions on Information and Systems*, Vol. E92-D, No.4, April 2009, pp. 584-593.