Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 681.324.06
DOI: https://doi.org/10.24867/25BE27Jankovic

IMPLEMENTACIJA PROTOKOLA ZA KOHERENTNOST KEŠ MEMORIJE

IMPLEMENTATION OF CACHE COHERENCE PROTOCOL

Jana Janković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prikazana je implementacija MESI protokola za koherentnost keš memorije, kao i ostatka multiprocesorskog sistema sastavljenog od RISC-V procesora i dva nivoa keš memorije. Pomoću bihevijalne simulacije testirano je ponašanje sistema u situaciji kada se iz drugog nivoa keš memorije izbacuje blok koji su modifikovala dva jezgra.*

Ključne reči: *Multiprocesorski sistemi, Koherentnost keš memorije MESI, FPGA*

Abstract – *This paper presents the implementation of the MESI protocol for cache coherence, as well as the rest of the multiprocessor system composed of RISC-V processors and two levels of cache memory. Behavioral simulation was used to test the behavior of the system in a situation where modified block is evicted from the second level cache.*

Keywords: *Multiprocessor Systems, Cache coherence, MESI, FPGA*

1. UVOD

Decenijama unazad mikroprocesorski sistemi doživljavali su eksponencijalan rast performansi zasnovan, kako na povećanju frekvencije signala takta, tako i na arhitekturalnim poboljšanjima. Dok su brzine čipova rasle zahvaljujući Murovom zakonu projektanti su se bavili ugrađivanjem novih tehnika za poboljšanja performansi procesora, među kojima su protočna obrada, predviđanje grananja, analiza toka podataka i spekulativno izvršavanje [1]. Protočna obrada uvodi paralelizam na nivou instrukcija podjelom njihovog izvršavanja na faze. Superskalarni procesori nastavljaju sa iskorišćavanjem paralelizma na nivou instrukcija omogućavajući paralelno izvršavanje nezavisnih instrukcija u više protočnih obrada.

Nažalost, postoje praktična ograničenja u napretku navedenih trendova razvoja. Dalja povećanja frekvencije, koja uzrokuju dalja povećanja dinamičke snage, čine problem disipacije snage teže rješivim. Među instrukcijama takođe postoji ograničena količina paralelizma koja se može iskoristiti do trenutka gdje zavisnost resursa onemogućava upotrebu više protočnih obrada.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vuk Vranjković, vanr. prof.

Navedena ograničenja i nejenjavajuće potrebe za boljim performansama dovode da, početkom 2000-ih godina dođe do neizbježnog prelaska na smještanje više prostijih procesora na jedan čip, odnosno pojave multiprocesorskih sistema.

Brzina prenosa podataka između glavne memorije i procesora nameće dodatni problem kroz činjenicu da ne može da prati nagli porast brzine procesora. Keš memorije rješavaju taj problem jer omogućavaju procesoru brži pristup podacima, usljed čega se u multiprocesorskim sistemima, radi postizanja prihvatljivih performansi, svakom procesoru dodjeljuje jedan ili dva privatna nivoa keš memorije. Uvođenje privatnih keš memorija u multiprocesorske sisteme sa dijeljenom memorijom stvara problem koherentnosti keš memorije. Rješenje ovog problema predstavljaju protokoli za koherentnost keš memorije.

2. PROBLEM KOHERENTNOSTI KEŠ MEMORIJE

Memorija u osnovi predstavlja skup lokacija koje posjeduju određenu vrijednost. Kada se desi operacija čitanja ona bi trebalo da vrati poslednju vrijednost upisanu na tu lokaciju. Sekvencijalni programi se oslanjaju na ovu osobinu kada koriste memoriju da komuniciraju vrijednost sa mjesta gdje se ona računa, do mjesta gdje se ona koristi [2]. U multiprocesorskim sistemima, u kojima se dijeljeni adresni prostor koristi za komunikaciju između procesa koji se izvršavaju na više različitih procesora, takođe je potrebno da pomenuta osobina bude zadovoljena. Privatne keš memorije mogu da predstavljaju problem jer procesi tada vide zajedničku memoriju kroz različite keš memorije. Naime, istovremeno, u različitim keš memorijama, može postojati više kopija jedinstvenih podataka. Sadržaj istih bio bi nedoslijedan kada bi procesorima bilo dozvoljeno da slobodno modifikuje isključivo svoje kopije podataka.

U osnovi dva zahtjeva moraju biti ispoštovana da bi se za multiprocesorski memorijski sistem reklo da je koherentan:

- Pročitana vrijednost sa memorijske lokacije mora odgovarati poslednjoj upisanoj vrijednosti na tu lokaciju. Ovaj zahtjev naziva se zahtjevom za propagacijom upisa (eng. write propagation).
- Izvršavanje više memorijskih operacija nad jednom memorijskom lokacijom mora biti viđeno u istom redoslijedu od strane svih procesora u sistemu. Ovaj zahtjev naziva se zahtjevom za serijalizacijom transakcija (eng. transaction serialization).

3. PROTOKOLI ZA KOHERENTNOST KEŠ MEMORIJE

Protokoli za koherentnost keš memorije obezbjeđuju komunikaciju između keš memorija, prilikom pristupa podacima, u cilju uspostavljanja koherentnosti.

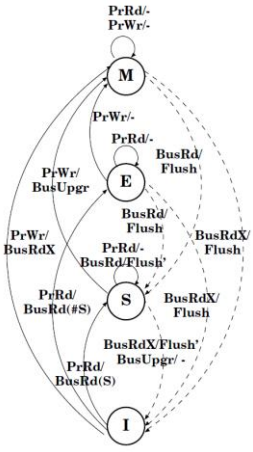
3.1. Podjela protokola za koherentnost keš memorije

Na osnovu načina na koji se izvodi propagacija upisa, dijele se na protokole sa invalidacijom upisa (eng. write invalidate) i protokole sa ažuriranjem upisivanja (eng. write update). Kod protokola sa invalidacijom upisa, kada jezgro želi da upiše novu vrijednost, inicira se transakcija koja će invalidirati kopije u drugim keš memorijama. Kod protokola sa ažuriranjem upisa, kada jezgro želi da upiše novu vrijednost, inicira se transakcija koja će ažurirati vrijednosti kopija u drugim keš memorijama.

Na osnovu toga da li se poruke šalju svim keš memorijama ili samo onim koje posjeduju podatak, protokoli se mogu svrstati u dvije grupe: protokoli sa direktorijumom (eng. directory protocols) i osluškujući protokoli (eng. snoopy protocols). Protokoli sa direktorijumom prate u kom kešu su prisutne kopije kojih blokova i te informacije prikupljaju u strukturi zvanoj direktorijum. Zahvaljujući direktorijumu poruke se mogu poslati samo onim keš memorijama koje posjeduju blok. Kod osluškujućih protokola poruka se šalje svim keš memorijama u sistemu. Osluškujući protokoli su pogodni za implementaciju u sistemima sa zajedničkom magistralom, obzirom da magistrala po prirodi stvari omogućava razglašavanje poruka. U takvim sistemima svaka keš memorija posjeduje kontroler koji osluškuje poruke koje se postavljaju na magistralu.

3.2. MESI protokol

MESI protokol je jedan od najčešće korištenih protokola za keš memorije sa odloženim ažuriranjem (eng. write back).



Slika 1. *Dijagram stanja MESI protokola [3]*

Dijagram stanja MESI protokola prikazan je na slici 1. Punom linijom prikazani su prelazi koji se dešavaju kao odgovor na zahtjeve procesora, dok su isprekidanom linijom prikazani prelazi koji predstavljaju odgovor na zahtjeve sa magistrale. Protokol posjeduje četiri stanja u kojima se mogu naći blokovi. Blok je u modifikovanom stanju (eng. modified, M) kada se njegov sadržaj razlikuje od onog u glavnoj memoriji i validan je u samo jednoj keš memoriji. Blok je u ekskluzivnom stanju (eng. exclusive, E) kada mu je sadržaj isti kao i u glavnoj memoriji i nalazi se samo u jednoj keš memoriji. Blok je u dijeljenom stanju (eng. shared, S) kada mu je sadržaj isti kao i u glavnoj memoriji i nalazi se u više keš memorija. Blok je u nevalidnom stanju (eng. invalid, I) kada mu je sadržaj nevažeći.Zahtjevi koji se mogu pojaviti na magistrali prilikom pristupa bloku prikazani su u tabeli 1.

Tabela 1. *Zahtjevi na magistrali*

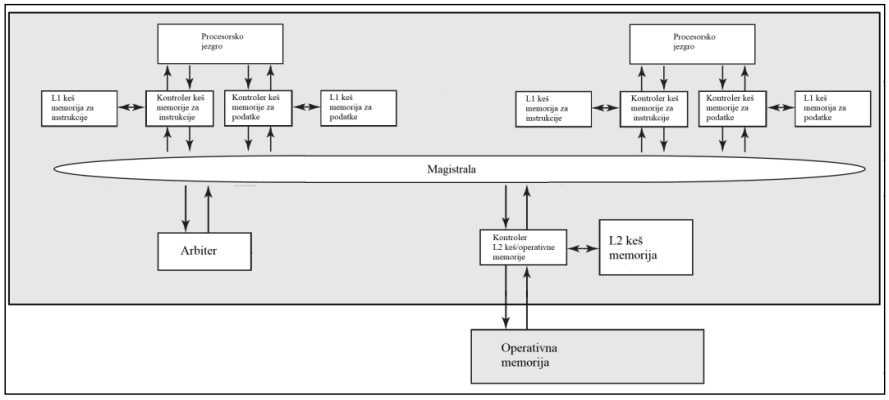
Naziv zahtjeva	**Opis**
PrRd	Zahtjev procesora za čitanjem bloka
PrWr	Zahtjev procesora za upisom u blok
BusRd	Zahtjev na magistrali za čitanjem bloka
BusRdx	Zahtjev na magistrali za ekskluzivnim čitanjem (upisom) bloka od strane procesora u čijem se kešu ne nalazi blok
BusUpgr	Zahtjev na magistrali za upisom u blok od strane procesora u čijem se kešu nalazi blok
Flush	Zahtjev na magistrali koji označava da se blok upisuje u glavnu memoriju
Flush’	Zahtjev na magistrali koji označava da je blok postavljen na magistrali. Uveden je za poboljšanje performansi i nije nužan za ispravan rad.

4.IMPLEMENTACIJA PROTOKOLA ZA KOHERENTNOST KEŠ MEMORIJE

4.1.Opis multiprocesorskog sistema

Organizacija multiprocesorskog sistema, za koji je implementiran protokol za koherentnost keš memorije, prikazan je na slici 2. Sistem se sastoji od više RISC-V procesorskih jezgara koji su povezani zajedničkom magistralom. Svako od jezgara posjeduje privatni prvi nivo keš memorije dok je drugi nivo dijeljen među njima. Prvi nivo je podijeljen na keš za instrukcije i keš za podatke. Svakoj od keš memorije je dodijeljen kontroler zadužen za uspostavljenje koherentnosti u sistemu. Kako je više uređaja povezano na magistralu, u sistemu se nalazi i arbiter koji reguliše pravo pristupa magistrali. Parametri sistema koji se mogu mijenjati su: broj procesorskih jezgara, veličina keš bloka i kapacitet keš memorija prvog i drugog nivoa.

Keš memorije prvog nivoa su 4-smjerne set-asocijativne memorije dok je keš memorija drugog nivoa 8-smjerna keš memorija. U svim keš memorijama sistema se koristi pseudo LRU algoritam zamjene zasnovan na binarnom stablu. Ovaj algoritam izabran je zbog dobrog kompromisa između performansi i broja bita potrebnih za njegovu implementaciju. Da bi se smanjila količina saobraćaja na magistrali za sve nivoe keš memorije korištena je metoda sa odloženim ažuriranjem nižeg nivoa hijerarhije uz koju ide metoda alociraj pri promašaju upisa. Keš drugog nivoa je inkluzivan sa keševima prvog nivoa zbog jednostavnije implementacije kontrolera. Protokol za koherentnost koji se koristi je MESI.



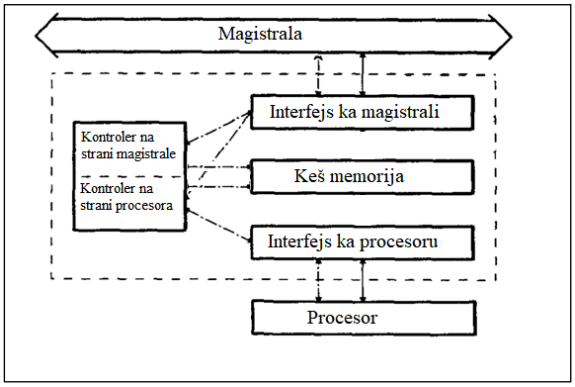
Slika 2. *Organizacija multiprocesorskog sistema [4]*

U keš memoriju za instrukcije se ne može upisivati, tako da za nju nije potrebno implementirati nijednu od metoda za ažuriranje nižih nivoa hijerarhije. Takođe, zbog toga, njen kontroler ne mora koristiti MESI protokol. Dovoljno je voditi evidenciju o tome da li je blok u memoriji validan ili ne.

U ovom projektu implementirano jezgro podržava RV32I skup instrukcija. Dodatno je implementirana i fence.I instrukcija, kako bi se postigla eksplicitna sinhronizacija memorije za instrukcije i podatke. Unutar procesora realizovana je protočna obrada sa pet faza.

4.2. Implementacija kontrolera koherentnosti keš memorije

Kontroler keš memorije za podatke mora istovremeno da odgovara na zahtjeve procesora i osluškuje dešavanja na magistrali. Zbog toga ga je najlakše posmatrati kao dva kontrolera jedan na strani procesora i jedan na strani magistrale [2,5]. Na slici 3 ilustrovan je takav kontroler. Kontroler na procesorskoj strani se ponaša u skladu sa zahtjevima procesora i zavisi od trenutnog stanja zahtjevang bloka u keš memoriji. Da bi se ispoštovao MESI protokol on mora posjedovati i pristup magistrali. Kontroler na strani magistrale osluškuje zahtjeve postavljene na magistrali i u skladu sa MESI protokolom i stanjem bloka u keš memoriji reaguje na njih. Keš memorija posjeduje dva porta kako bi joj istovremeno mogla pristupiti oba kontrolera.

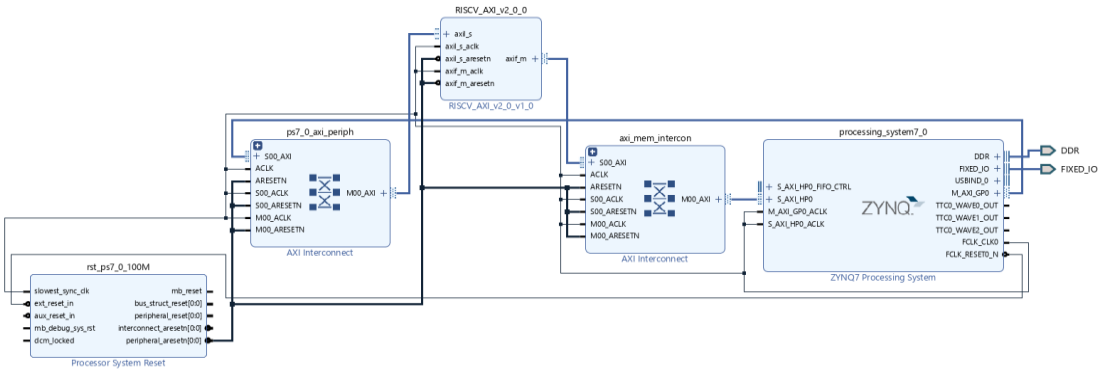


Slika 3. *Unutrašnja organizacija kontrolera [5]*

Kada kontroler na strani procesora utvrdi u kom je stanju zahtjevani blok u keš memoriji on će od arbitera tražiti dozvolu da postavi odgovarajući zahtjev na magistralu. Nakon što dobije dazvolu i postavi zahtjev na magistralu kontroler na strani procesora čeka da kontroleri drugih keš memorija u sistemu odgovore na zahtjev. Kada dobije odgovor, kontroler može promijeniti stanje bloka i odgovoriti na zahtjev procesora. Može se desiti da, dok kontroler na strani procesora čeka na dozvolu pristupa magistrali, kontroler na strani magistrale promijeni stanje bloka kom procesor želi da pristupi. Kontroler na procesorskoj strani mora biti u mogućnosti da detektuje tu promjenu stanja i da u skladu sa njom promijeni zahtjev koji se postavlja na magistralu.

Problem može nastati i kada i jedna i druga strana kontrolera istovremeno žele da promijene stanje istog bloka. Te situacije se rijetko dešavaju zbog potrebe postavljanja zahtjeva na magistralu prilikom prelaska iz jednog stanja u drugo. Primjer za to, u MESI protokolu, bi bila situacija u kojoj procesorska strana želi da modifikuje blok u E stanju i promijeni mu stanje u M, dok zahtjev sa magistrale želi da promijeni stanje tog bloka u S ili I. Kontroleru na strani magistrale se tada daje prednost jer je on prethodno taj blok postavio na magistralu i procesorskoj strani se ne može dozvoliti da ga modifikuje.

Kada se odgovara na zahtjev sa magistrale, često je potrebno prvo postaviti blok podataka na magistralu i zatim mu promijeniti stanje. Može se desiti da u trenutku kada se javi zahtjev za koji je potrebno prenijeti blok, procesor želi da modifikuje isti, bez potrebe za pristupanjem magistrali. Procesoru se može omogućiti da tada modifikuje blok, međutim, svaki naredni upis za vrijeme trajanja prenosa, zbog potrebe da sistem bude koherentan, mu ne može biti omogućen. To se postiglo uvođenjem signala *ProcWants* i *SnoopHas*. Kontroler na strani magistrale provjerava da li je signal *ProcWants* postavljen na 1 i drži vrijednost signala *SnoopHas* na 0. Ukoliko jeste, čeka da vrijednost signal postane 0, nakon čega u sljedećem taktu prelazi u stanje u kom postavlja vrijednost signala *SnoopHas* na 1 i započinje slanje bloka. Kontroler na strani procesora prvo provjerava vrijednost signala *SnoopHas*. Ako je ona 0, vrijednost signala *ProcWants* postaje 1, u suprotnom je 0. Na taj način procesor dobija prednost, kada prije početka prenosa želi da upiše novu vrijednosti u blok. Čim se prekine upis (*ProcWants* postane 0), kontroler na strani magistrale od sljedećeg takta započinje prenos i ne dozvoljava dalji upis u blok.



Slika 4. *Blok dizajn sistema*

4.3 Simulacija sistema

Pomoću bihevijalne simulacije testirano je ponašanje sistema u situaciji kada se modifikovani blok izbacuje iz drugog nivoa keš memorije. Sistem koji se testira sadrži dva jezgra, veličina keš bloka je 64B, kapacitet keš memorije prvog nivoa je 1KB i kapacitet keš memorije drugog nivoa je 4KB. Programi koji su namjenjeni da se izvršavaju na procesorima smješteni su na različitim memorijskim lokacijama. Procesori moraju posjedovati informaciju o početnoj adresi programa koji izvršavaju. U sistem se, zbog toga, uvode registri u koje se mogu, preko AXI-Lite interfejsa, upisati početne adrese programa i registar za signal dozvole takta.

```
lui a0, 6
addi a1, a0, 8
sw a1, 0(a0)
lui a0, 5
lw a0, 0(a0)
lui a0, 4
lw a0, 0(a0)
lui a0, 3
lw a0, 0(a0)
lui a0, 1
lw a0, 0(a0)
```

```
lui a0, 6
lw a1, 0(a0)
addi a1, a1, 4
sw a1, 4(a0)
lui a0, 7
lw a0, 0(a0)
lui a0, 8
lw a0, 0(a0)
lui a0, 9
lw a0, 0(a0)
lui a0, 10
lw a0, 0(a0)
```

Slika 5. *Programi za testiranje izbacivanja modifikovanog bloka iz keš memorije drugog nivoa*

Za testiranje ponašanja sistema, u situaciji kada se modifikovan blok izbacuje iz keš memorije drugog nivoa, koriste se programi prikazani u kodnom listingu, slika 5. U ovim jednostavnim asemblerskim kodovima jedan procesor prvo pristupa lokaciji 0x6000 i na nju upisuje vrijednost 0x6008, dok drugi procesor prvo čita sadržaj lokacije 0x6000 uvećava je za 4 i upisuje na lokaciju 0x6004. Procesori zatim čitaju sadržaj memorijskih lokacija različitih blokova, sve dok ne dođe do potrebe da se jedan blok izbaci iz keš memorije drugog nivoa kako bi se novi upisao. Blok koji se mora izbaciti je prvi blok koji su modifikovala oba procesora. Prilikom izbacivanja bloka, na osnovu vrijednosti koje se upisuju nazad u operativnu memoriju može se zaključiti da je u sistemu postignuta koherentnost

4.4 Implementacija sistema na Zybo razvojnoj ploči

Sistem testiran u simulaciji upakovan je u IP jezgro. Zapakovano jezgro posjeduje *AXI-Lite Slave* i *AXI-Full Master* interfejse. IP jezgro je povezano sa Zynq procesorskim sistemom kao na slici 4. Nad sistemom sa slike 4 izvršene su sinteza i implementacija. Maksimalna frekvencija rada sistema je 71.42MHz. Kritičnu putanju predstavlja putanja računanja adrese za pristup memoriji u trećoj fazi protočne obrade. Putanja prolazi kroz multipleksere koji određuju koji signali se propuštaju na ulaze ALU jedinice, zatim kroz ALU jedinicu i stiže na adresni ulaz memorije za podatke, tagove i LRU stanja. Najveće kašnjenje je kroz kanale za rutiranje je 10.855ns; kašnjenje kroz logičke elemente je 2.420ns. Sistem se na kraju testira u Vitis alatu, pomoću C aplikacije.

5. ZAKLJUČAK

U radu su prikazani bazni dijelovi mplementacija MESI protokola za koherentnost keš memorije, i ostatka multiprocesorskog sistema, sastavljenog od RISC-V procesora i dva nivoa keš memorije. Rezutati testiranja ponašanja sistema pomoću bihevijalne simulacije se poklapaju sa rezultatima nakon testiranja sistema u Vitis alatu, pomoću C aplikacije.

Potencijalno poboljšanje performansi navedenog sistema moglo bi ići u smjeru korištenja magistrale dijeljenih transakcija i/ili primjenom principa ekskluzije.

6. LITERATURA

[1] W. Stallings, "*Computer Organization and Architecture: Designing for Performance*", 9th ed. Boston, MA: Pearson, 2013.

[2] D. Culler, J. P. Singh, and A. Gupta, "*Parallel Computer Architecture: A Hardware/Software Approach*", San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[3] T. Suh, "*Integration and evaluation of cache coherence protocols for multiprocessor socs*", 2006.

[4] V. Nagarajan, D. J. Sorin, M. D. Hill, and D. A. Wood, " *A Primer on Memory Consistency and Cache Coherence*", 2nd ed. Morgan & Claypool Publishers, 2020

[5] R. H. Katz et al., "*Implementing a cache consistency protocol*", ACM SIGARCH Computer Architecture News, vol. 13, no. 3, pp. 276-283, 1985.

Kratka biografija:

Jana Janković rođena je u Bijeljini 1999. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Mikroračunarska elektronika, Embeded sistemi i algoritmi, odbranila je 2022. god
kontakt: janajankoviic@gmail.com