Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: (Upisuje redakcija, ostaviti ovaj red)
DOI: (Upisuje redakcija, ostaviti ovaj red)

IMPLEMENTACIJA PODSISTEMA SKRIVENE MEMROIJE ZA RISC-V PROCESOR

IMPLEMENTATION OF CACHE SUBSYSTEM FOR RISC-V PROCESSOR

Đorđe Mišeljić, Vuk Vranjković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je modelovan podsistem skrivene (keš, eng. cache) memorije za RISC-V procesor. Model je pisan u VHDL jeziku te je ciljan za soft-core primenu na FPGA uređajima. Model se sastoji iz dva nivoa keš hijerarhije: prvi nivo je direktno preslikan i razdeljen, dok je drugi N-smerno set asocijativan i unificiran. Model je parametrizovan, te korisnik ima mogućnost da bira kapacitet memorija kao i asocijativnost. Sistem je zajedno sa jednostavnim RISC-V procesorom prvo simuliran pomoću Vivado alata a zatim upakovan u IP jezgro, implementiran i testiran na Zybo razvojnoj ploči.*

Ključne reči: *Skrivena memorija, RISC-V, FPGA, Zybo*

Abstract – *This paper follows a modeling of a cache subsystem for the RISC-V processor. The model was written in VHDL language and is targeted at soft-core applications on FPGA devices. The model is composed of two levels of cache hierarchy: the first level is directly mapped and split, while the second level is n-way set associative and unified. The model is parametrized, so user can choose cache size as well as associativity. A simulation of a simple RISC-V processor core with a cache subsystem was done in the Vivado development tool, after which it was packaged into an IP core, implemented, and tested on the Zybo development board.*

Keywords: *Cache memory, RISC-V, FPGA, Zybo*

1. UVOD

Poslednjih par decenija, dve arhitekture skupa instrukcija procesora (*eng. Instruction Set Architecture*) su kontrolisale svetsko tržište: x86 i ARM. Obećavajući faktor za promenu trenutnog stanja u hardverskoj industriji je nova, besplatna i otvorena arhitektura pod nazivom RISC-V [1]. Krećući se od osnovnog seta instrukcija, moguće je dodati proširenja koje su specijalno dizajnirana za različite komercijalne i naučne svrhe. Sa ovim na umu, moguće je da RISC-V nađe primenu kako u oblasti visokih performansi tako i u oblasti uređaja male potrošnje energije, te na taj način napravi otvoreni standard za hardver. Prednost ovoga je razvoj softvera koji će biti kompatibilan i raditi na svim sličnim RISC-V jezgrima [2].Posebna oblast od interesa za RISC-V arhitekturu su FPGA (eng. *Feild Programmable Gate Array*) uređaji.

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

Ukoliko neko implementira RISC-V procesorsko jezgro na FPGA čipu, često obezbedi i RTL (eng. *Register Transfer Level*) izvorni kod. Budući da RISC-V ima i punu softversku podršku Linux organizacije, moguće je bez ikakvih izmena koda, besplatno i legalno podići operativni sistem baziran na Linux-u, na bilo kojoj FPGA razvojnoj ploči [3]. Stoga, postoji velika zainteresovanost za dizajn *soft-core* RISC-V procesora, za koje se veruje da će verovatno zameniti ulogu Xilinx-ovog MicroBlaze procesora koji je do sada bio jedno od retkih rešenja za podizanje operativnog sistema na FPGA čipu. Ono što će izdvojiti RISC-V u odnosu na MicroBlaze je softverska podrška koja je uvek bila slaba tačka *soft-core* procesora zbog malog broja korisnika i odžavaoca softvera.

Ne sme se zaboraviti da je brzina pristupa memoriji i dalje usko grlo u performansama procesora. Svaki moderan procesorski čip na sebi ima barem dva nivoa keš memorije, koja zbog težnje da bude što veća, zauzima veliki deo ukupne površine čipa. Dakle, benefiti skrivenih memorija se ne smeju ignorisati, čak i u oblastima procesiranja sa malom potrošnjom energije. Kao i kod ASIC (eng. *Application-Specific Integrated Circuit*) implementacije, *soft-core* procesori na FPGA čipovima mogu iskoristiti različite tipove memorijskih resursa koji su na raspolaganju dizajneru, kao spregu između procesorske logike i operativne DDR RAM (eng. *Dual Data Rate Random Access Memory*). Ovaj rad će predstaviti jedan način iskorištenja konfigurabilnih memorijskih resursa koji su prisutni na svakom savremenom FPGA čipu, kako bi se implementiralo keširanje i zadovoljila memorijska zahtevnost RISC-V procesora.

2. MODEL KEŠ PODSISTEMA

Od osnivanja ideje o keširanju podataka, svaki savremeni memorijski sistem se deli na više memorijskih nivoa, gde se manje a brže memorije postavljaju što bliže procesoru, dok su veće a sporije memorije na periferiji [4]. Niži nivoi su manjeg kapaciteta, optimizovanije fabrikacije tranzistora (veća brzina, potrošnja i površina) te su bliži procesoru za manje kašnjenje na linijama. Uz pomoć dodatnog hardvera (keš kontrolera) proces prebacivanja memorije između različitih nivoa keševa i operativne memorije se izvršava automatski, bez potrebe intervencije programera.

Za datu memorijsku hijerarhiju, gde t_i predstavlja tehnološki zavisno kašnjenje pristupa memoriji na nivou hijerarhije i (krećući od strane procesora), realno vreme označeno sa T_i je veće jer zavisi i od ostalih nivoa hijerarhije memorije. Vreme T_i zavisi od udela pogodaka h_i (eng. *hit rate*) i promašaja m_i (eng. *miss rate*). Ovi

brojevi govore koliko se često traženi podatak nalazi u memoriji na nivou *i*. Ukoliko se desi pogodak, procesor može da preuzme podatak iz memorije na nivou *i*, a u suprotnom se prelazi na nivo *i*+1. Na sličan način se redom ispituju nivoi *i*+2...*N* dok se ne pronađe željeni podatak. Iz ovoga proizlaze formule 1, 2 i 3 [5].

$$h_i + m_i = 1 \quad (1)$$

$$T_i = h_i \cdot t_i + m_i \cdot (t_i + T_i + 1) \quad (2)$$

$$T_i = t_i + m_i \cdot T_i + 1 \quad (3)$$

Formula br. 3 se dalje može rekurzivno rastavljati zamenom člana T_{i+1} sa zavisnošću sa nivoom *i*+2. Za poslednji nivo hijerarhije memorije važi da je h_N=1 dok je m_N=0 jer se podatak mora nalaziti u njoj, te je $T_N = t_N$. Cilj pri dizajniranju sistema za keširanje je dobiti željeno vreme T_i unutar dozvoljenog budžeta.

2.1. Tipovi memorija na Zybo ploči

Za implementaciju keširanja na Zybo razvojnoj ploči se mogu iskoristiti dva tipa memorijskih resursa koji su prisutni u programabilnoj logici:

Blok RAM: postoji 60 blokova, gde se svaki može konfigurisati kao dvopristupna memorija veličine 36Kb ili dve jednopristupne memorije veličine 18Kb [6]. Čitanje memorijske lokacije se vrši na rastuću ivicu sinhronizacionog signala. Može se uključiti izlazni registar, čime se unosi dodatan takt kašnjenja pri čitanju, ali se dobija manje *clk-to-output* kašnjenje. Ukupno je na raspolaganju 2.1Mb ovog tipa memorije.

Distribuirani (*LUT, Lookup Table*) RAM: Od prisutnih 17,600 *slice* modula, 6000 su tipa *slicem*, dok je ostatak tipa *slicel* [7]. Svaki *slicem* modul se može konfigurisati kao 64 bita brze RAM memorije [8]. Čitanje iz ove memorije je asinhrono, ali se može dodati izlazni registar za bolju *clk-to-output* karakteristiku. Ukupno je na raspolaganju 375Kb ovog tipa memorije. Brža a manja distribuirana (LUT) RAM će se iskoristiti za prvi nivo, dok sporija a veća blok RAM će se iskoristiti za drugi nivo keša.

2.2. Direktno preslikan keš

Podaci se između nivoa memorijske hijerarhije prenose u blokovima. Veličina bloka od *B* bajtova, znači da će nižih $b=ceil(log_2 B)$ bita u adresi referencirati bajt u bloku (eng. *byte in block, bib*). Sa veličinom keša od *C* bajtova, širina adrese keš memorije će biti $c= ceil(log_2 C)$ bita. Nižih *c* bita u adresi traženog podatka će indeksirati lokacije u kešu. Kod direktno preslikanog keša, kada se blok preuzima iz operativne memorije (eng. *fetch*), on se smešta na memorijske lokacije u kešu koje se poklapaju sa donjih *c* bita adrese. Može se zaključiti da postoji $2^{(31-c)}$ različitih blokova koji se mogu nalaziti na istoj lokaciji u kešu. Kako bi se znalo koji je blok trenutno na nekoj lokaciji u kešu, kada se on preuzima operativne memorije, gornjih (31-*c*) bita pod nazivom "tag" se čuva u pomoćnoj memoriji za čuvanje tagova (eng. *tag store*). Ova memorija se često implementira u istoj tehnologiji izrade kao i keš. U ovom radu, prvi nivo keša je implementiran kao direktno mapiran keš sastavljen od LUT RAM-a.

2.3. Set asocijativan keš

Najveća mana keširanja sa direktnim preslikavanjem je da se lako može napraviti sekvenca pristupa memoriji koja proizvodi udeo pogodaka jednak nuli - naizmeničan pristup dvema memorijskim lokacijama sa istim indeksom. Kako bi se ovo izbeglo, keš se pravi da bude N-set asocijativan. Keš se podeli u N manjih jednakih delova, svaki sa svojom memorijom za čuvanje tagova, te u istom trenutku može da čuva N podataka sa istim indeksom. Sa većom asocijativnošću se udeo pogodaka povećava na račun dodatne logike i brzine rada. Potrebno je imati N komparatora za poređenje tagova, te je potrebno pomoću dodatne kombinacione logike proslediti podatak iz jednog od N smerova. U ovom radu je implementiran parametrizovani N-smerni keš, sastavljen od blok RAM modula na FPGA čipu.

2.4 Polisa evikcije

Kada se keševi inicijalizuju, svi blokovi su nevalidni, te se N smerova popunjava redno. Ukoliko su svi blokovi u N smerova validni, onda se usvaja set pravila na osnovu kojih se odlučuje koji od blokova će biti zamenjen. Najočiglednija polisa je izbaciti najdavnije korišten blok (eng. *Least Recently Used, LRU*). Implementacija ove polise postaje problem za keševe sa većom asocijativnošću. U praksi se pokazuje da udeo pogodaka zavisi od programa na kojima se testira, te je prosečan udeo pogodaka za sličan za LRU i nasumičan odabir bloka [9]. Iz ovog razloga se pribegava tehnikama koje predstavljaju kombinaciju LRU i nasumičnog algoritma koje se zovu *Pseudo LRU* polise. U ovom radu je za N-smerni asocijativan keš drugog nivoa implementirana pseudo LRU polisa pod imenom "Žrtva - Sledeća žrtva" (eng. *Victim - Next victim*) [10]. Od N blokova u asocijativnom kešu samo se prate dva bloka: jedan označen kao žrtva (eng. *victim*) i jedan označen kao sledeća žrtva (eng. *next victim*), dok su svi ostali blokovi obični (*ordinary*).

2.5 Polisa upisa

Kada procesor izvrši naredbu upisa podatka u memoriju, te se desi pogodak, on promeni sadržaj nekog bloka u najnižem nivou keša. Pitanje je: kada ažurirati sledeći nivo keša sa novim podatkom? Ukoliko se keš dizajnira tako da se promeni sadržaj bloka samo najnižeg nivoa, dok blok zadržava prethodnu vrednost u sledećem nivou keša, sledeći nivo će biti ažuriran tek kada se blok eviktuje iz prethodnog nivoa. Ovakav keš se naziva "upis-nazad" (eng. *write-back*) keš. Druga bitna odluka vezana za upise u keševe je: da li se alociraju keš blokovi kada se desi promašaj pri instrukciji upisa? Pri metodi "alociraj pri promašaju upisa" (eng. *allocate on write miss*), keš blok se preuzima iz keša višeg nivoa ili operativne memorije, te se nakon toga izvrši upis. U ovom radu će postojati dva nivoa keša gde će oba biti metode "upis-nazad" i "alociraj pri promašaju upisa".

2.6 Polisa razdeljenosti

Većina savremenih procesora zahteva da se na svaku rastuću ivicu takta iz memorije preuzme instrukcija koju je potrebno dekodovati, a sem toga, potencijalno dodatan pristup memoriji ukoliko se radi o instrukciji upisa ili čitanja iz memorije. Stoga se keš prvog nivoa može implementirati kao jedna dvopristupna ili kao dve fizički odvojene memorije (za instrukcije i podatke). Prvi način

se naziva ujedinjeni (eng. *unified*), a drugi se naziva razdeljeni (eng. *split*) keš.

Većina modernih procesora ima implementiranu protočnu obradu podataka, te se pri postavljanju komponenti na čip i optimizaciji rutiranja, desi da je logika faze za prihvat instrukcija fizički udaljena od faze za upis/čitanje podataka. Razdeljeni keš omogućava da se keševi za čuvanje instrukcija i podataka smeste na čipu blizu faze protočne obrade kojoj su potrebni. Na FPGA čipovima veliki problem prestavlja rutiranje, koje je mnogo ograničenije nego u ASIC implementaciji - zbog mnogo manjeg izbora za postavljanje komponenti te ograničenog broja kanala za rutiranje. Na rutiranju se često gubi veći deo performansa implementirane logike. Iz ovih razloga je dobar izbor da se prvi nivo keša implementira kao razdeljeni, a drugi nivo kao unificiran keš. Prvi nivo keša pokazuje najbolje rezultate kada su keševi za instrukcije i podatke jednake veličine [11].

2.7 Polisa inkluzije

Kod inkluzivnog keša, prisusvo bloka u nižem nivou keša implicira postojanje kopije istog bloka u višem nivou keša. Inkluzivan keš žrtvuje ukupan kapacitet keša zbog dobiti na jednostavnosti implementacije keš kontrolera. U ovom radu će dva nivoa keša biti inkluzivna.

3. IMPLEMENTACIJA SISTEMA NA ZYBO PLOČI

Implementacija sistema je izvršena u Vivado alatu kompanije Xilinx. Ciljana razvojna ploča je Zybo, kompanije Digilent, koja na sebi ima Zynq-7000 sistem na čipu (eng. *System on Chip, SoC*). Ploča poseduje dva DDR3 memorijska čipa, koja prave 32-bitni interfejs ka memoriji kapaciteta 512MB i propusnim opsegom 1050Mbps. Na Zynq SoC-u postoji već ugrađen memorijski kontroler sa 8 DMA (eng. *Direct Memory Access*) kanala za direktan pristup memoriji. Za potrebe testiranja, sistem za keširanje sa RISC-V procesorom je implementiran u programabilnoj logici, te je jedan od memorijskih kanala iskorišten za komunikaciju sa operativnom memorijom. Interfejs je tipa AXI3-Full širine 64 bita.

3.1. Pakovanje IP jezgra

Kako bi bilo moguće uvesti kontrolne i statusne signale za processor, kao i omogućiti da keš sistem pravilno komunicira sa DDR memorijskim čipom, moraju se modelovati *AXI-Lite Slave* i *AXI-Full Master* interfejsi. 32-bitni RISC-V processor koji podržava osnovni set instrukcija RV32I, je zajedno sa opisanim podsistemom za keširanje oklopljen u IP (eng. *Intelectual Property*) jezgro. Upakovana su i dva dodatna modula koja implementiraju potrebne AXI interfejse te služe kao sprega za komunikaciju sa Zynq SoC-om.

3.2. Blok dizajn u Vivado alatu

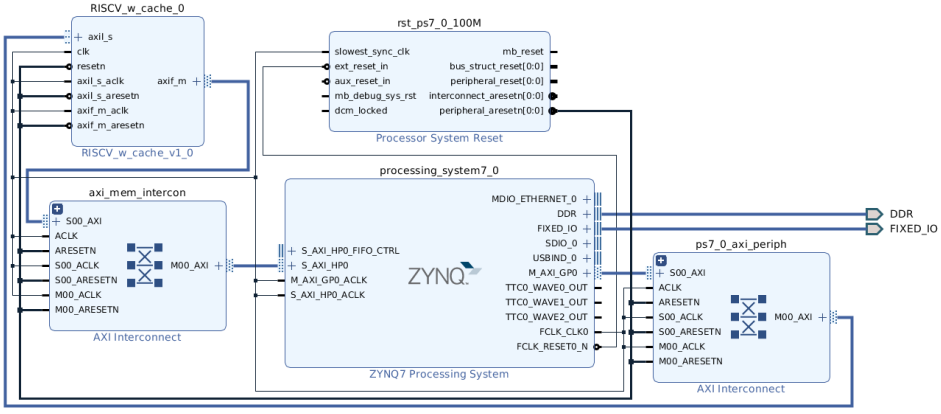
Prethodno opisano IP jezgro je povezano sa Zynq SoC-om kao na slici 1. 32-bitni AXI4 Master interfejs jezgra RISCV_AXI je povezan sa 64-bitnim AXI3 Slave interfejsom memorijskog kontrolera preko interkonekt modula. Interkonekt pravi spregu između AXI4 i AXI3 protokola, te vrši pakovanje podataka u 64-bitni interfejs. Dodatno je omogućena funkcija interkonekta za unutrašnji FIFO (eng. *First In First Out*) bafer, kako bi potencijalna memorijska kašnjenja unutar transakcije bila neprimetna iz tačke gledišta RISCV_AXI jezgra. Sistem je implementiran i testiran sa sledećom konfiguracijom parametara:

- *BLOCK_SIZE* = 64; - veličina keš bloka je 64B.
- *LVL1_CACHE_SIZE* = 1024*;* - veličine keševa prvog nivoa su 1KB (za instrukcije i podatke).
- *LVL2_CACHE_SIZE* = 4096*;* - veličina svakog smera u drugom nivou keša je 4KB.
- *ASOCIATIVITY* = 4; - drugi nivo poseduje četri smera u set asocijativanom kešu.
- *TS_BRAM_TYPE = HIGH_PERFORMANCE;* - koristi se dodatan izlazni registar u memorijama za čuvanje tagova drugog nivoa.

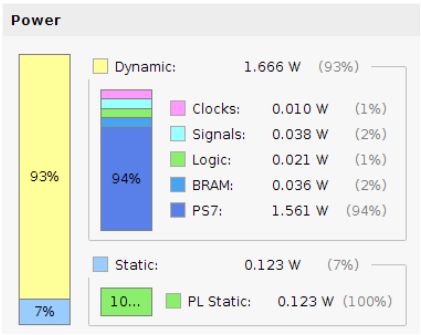
Frekvencija rada sistema je podešena na 100MHz. Sinteza i implementacija su uspešno završene. Iskorištenost resursa nakon implementacije se može videti u tabeli 1, a potrošnja energije na slici 2.

Tabela 1. *Iskorištenost resursa*

Resurs	Zauzeto	Dostupno	Zauzeto %
LUT6	5063	17600	28.77
LUTRAM	512	6000	8.53
FF	3308	35200	9.40
BRAM	8	60	13.33
BUFG	1	32	3.13



Slika 1. *Blok dijagram sistema na Zybo ploči*



Slika 2. *Potrošnja energije*

3.3. Testiranje u Vitis alatu

Za svrhe testiranja je u blok dizajnu prikazanom na slici 1, dodat modul *System ILA* (eng. *Integrated Logic Analyzer*). Ovaj modul je povezan na AXI Full interfejs RISCV_AXI modula, te koristi blok RAM da zabeleži transakcije pri testiranju na Zybo ploči. Na ovaj način možemo proveriti da li se transakcije koje inicira sistem za keširanje poklapaju sa zahtevima programa, te dobiti informaciju o realnim kašnjenjima pri pristupu memoriji.

ILA modul je podešen da detektuje transakcije čitanja iz memorije te da u tom trenutku sačuva vrednosti signala na AXI interfejsu. Na osnovu zabeleženih vrednosti se pokazuje da se od zahteva čitanja bloka, do čitanja poslednjeg podataka iz DDR memorijskog čipa utroši približno 42 mašinska ciklusa. Od trenutka kada se na AXI Full interfejsu postavi adresa zahtevanog bloka, do prijema prvog podatka prođe 26 ciklusa, koji se gube zbog sporog odziva DDR čipa. Ostalih 16 ciklusa je potrebno da se blok od 64B prenese preko magistrale širine 32 bita. Ukoliko se ovo uporedi sa prenosom bloka iz drugog nivoa keša što zahteva 16 mašinskih ciklusa za prenos podataka sa dodatna 3 takta za proveru tagova i ažuriranje pomoćnih bita, očigledno je zašto je memorijski sistem sa više nivoa keševa neophodan za dobre performanse procesora. Može se takođe zaključiti da za manje blokove, odnos korisnih ciklusa u odnosu na utrošene u čekanju na DDR čip, naglo opada. Slični rezultati se dobijaju za upis bloka podataka u DDR čip kada se ILA podesi u modu za detekciju transakcija upisa bloka podataka. Od inicijalizacije transakcije upisa implementaciju memoriju, do odgovora memorije da je upis izvršen uspešno (eng. *response*), utroši se 38 mašinskih ciklusa.

4. ZAKLJUČAK

Kako bi se premostila razlika između brzine rada procesora i tipičnog DDR3 čipa, u *soft-core* primeni RISC-V arhitekture, mogu se iskoristiti već postojeći memorijski resursi na FPGA ploči kako bi se implementirao jednostavan a efikasan sistem za keširanje. Distribuirani RAM može da obezbedi prvi nivo keša malog kapaciteta ali dovoljno brzog odziva da zadovolji zahteve pristupa memoriji bez zaustavljanja protočne obrade. Nasuprot tome, blok RAM može da obezbedi dovoljno velik kapacitet keša kako bi se aktivan set podataka aplikacije čuvao na FPGA čipu. HDL model koji je predstavljen u radu je samo jedan način realizacije sistema za keširanje. Implementirana inkluzivnost, pamćenje prisustva blokova prvog nivoa u pomoćnim bitima, kao i slobodan port u memorijama drugog nivoa keša izgleda kao dobro rešenje za višejezgarni *soft-core* processor, jer predstavlja odličnu osnovu za laku implementaciju MOESI (*Modified, Owned, Exclusive, Shared, Invalid*) ili sličnog algoritma za održanje koherencije između jezgara [12].

4. LITERATURA

[1] K. Asanović, D.A. Patterson, “Instruction Sets Should Be Free: The Case For RISC-V”, EECS Dept., U.C. Berkley, CA. UCB/EECS-2014-146. Aug.6, 2014. Dostupno: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.pdf

[2] https://riscv.org/why-risc-v/ (pristupljeno u avgustu 2020.)

[3] https://www.linuxfoundation.org/the-linux-foundation/2018/11/the-linux-foundation-and-risc-v-foundation-announce-joint-collaboration-to-enable-a-new-era-of-open-architecture/ (pristupljeno u septembru 2020.)

[4] M. V. Wilkes, “Slave Memories and Dynamic Storage Allocation”. IEEE Trans. Electron. Comput., pp. 270-271, Apr. 1965.

[5] J. C. Hoe, Predavanje, “Memory Technology and Organization” Carnegie Mellon Univ., Pittsburgh, PA, Mar. 2020. Dostupno: http://users.ece.cmu.edu/~jhoe/course/ece447/S20handouts/L14.pdf

[6] Xilinx, *7 Series FPGAs CLB User Guide (UG474)* Dostupno: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[7] Xilinx, *Zynq-7000 SoC Data Sheet: Overview (DS190)*, 2018. Dostupno: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[8] Xilinx, *7 Series FPGAs Memory Resources User Guide (UG473)*, 2018. Dostupno: https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf

[9] L. Belady, “A Study of Replacement Algorithms for a Virtual-Storage Computer”. IBM Syst. J., no. 5, pp.78-101, 1966.

[10] O. Mutlu, Predavanje, “Caches”. Carnegie Mellon Univ., Pittsburgh, PA, Feb. 2015. Dostupno: https://course.ece.cmu.edu/~ece447/s15/lib/exe/fetch.php?media=onur-447-spring15-lecture18-caches-afterlecture.pdf

[11] J. Bell, D. Casasent, C.G. Bell, “An Investigation of Alternative Cache Organizations”. IEEE Trans. Comput., pp. 346-351, Apr. 1974.

[12] J. Hennessy, D. Patterson, *Computer Architecture : A Quantitative Approach*, 6th ed., San Francisco, CA: Morgan Kaufmann Publishers Inc., 2017, pp 377-388.

Kratka biografija:

Đorđe Mišeljić rođen je na Cetinju (Crna Gora) 1995. god. Završio je opštu gimnaziju u SMŠ Ivan Goran Kovačić 2014. god u Herceg Novom. Bečelor rad na Fakultetu tehničkih nauka, usmerenje Embeded sistemi i algoritmi, odbranio je 2018. godine, kada je izabran u zvanje saradnik u nastavi. Kontakt: djordjemiseljic@uns.ac.rs