Realizacija CQ Ethernet komutatora na NetFPGA razvojnoj platformi

Nikola Ljumović, Danilo Mišović, Milutin Radonjić, *Member, IEEE*, Igor Radusinović, *Member, IEEE*

***Sadržaj* — U ovom radu je prikazana hardverska realizacija Ethernet komutatora implementiranog na NetFPGA razvojnoj platformi. Komutator je realizovan upotrebom *crossbar* komutacione matrice sa baferima u ukrsnim tačkama. Opisani komutator na efikasan način vrši obradu i prosleđivanje Ethernet frejmova koji na njegove portove pristižu brzinom od jednog gigabita u sekundi. Značaj ove realizacije se ogleda u mogućnosti testiranja i nadogradnje pomenute arhitekture mrežnog uređaja, što će omogućiti kvalitetan nastavak do sada sprovedenog rada na analizi performansi CQ komutatora, kroz poređenje simulacionih i analitičkih rezultata sa rezultatima koji se dobijaju u realnom mrežnom okruženju.**

***Ključne riječi* — crossbar, komutator, lookup table, NetFPGA, scheduler, switching fabric.**

I. Uvod

JEDAN od ključnih trendova savremenih telekomunikacija je ekspanzija Interneta u poslednjih petnaestak godina. Kao neminovnost javlja se značajan porast intenziteta saobraćaja u svim njegovim segmentima, što konstantno nameće potrebu za kreiranjem sve bržih i efikasnijih mrežnih uređaja. Istraživanje na ovom polju i realizovanje ovih uređaja je ranije predstavljalo kompleksan i vremenski zahtjevan proces koji se odvijao isključivo u razvojnim institutima velikih proizvođača telekomunikacione opreme. Međutim, danas istraživačima na raspolaganju stoji efikasan metod kreiranja i testiranja mrežnih uređaja koji se ogleda u primjeni FPGA (*Field Programmable Gate Array*) integrisanih kola.

FPGA pruža korisnicima mogućnost brzog konfigurisanja i realizovanja ogromnog broja logičkih funkcija korišćenjem programskog jezika za opisivanje hardvera (Verilog, VHDL) [1]. Zbog toga, danas FPGA uređaji (razvojne pločice) sve više predstavljaju nezaobilazan element u dizajniranju i projektovanju digitalnih sistema različitih namjena.

Jedna od takvih razvojnih platformi, kreiranih upravo za potrebe istraživanja na polju mrežnih uređaja, je NetFPGA pločica [2], osmišljena i kreirana na Stanford Univerzitetu u SAD. Njena velika popularnost potiče od činjenice da ova otvorena razvojna platforma u velikoj mjeri olakšava istraživanje na ovom polju, pružajući mogućnost kreiranja naprednih, korisnički definisanih, mrežnih uređaja, unapređenja postojećih i razvoja novih mrežnih protokola. Odsustvom ovakvog rješenja, istraživanja u ovoj oblasti bi bila privilegija malog broja istraživača u velikim naučnoistraživačkim laboratorijama, ili bi bila svedena na rad sa zatvorenim mrežnim sistemima, čiji bi funkcionalnost i dizajn bili u potpunosti određeni od strane proizvođača.

Da se pri radu na svakom novom projektu ne bi polazilo od samog početka, istraživači sa Stanford-a su razvili tzv. referentnu arhitekturu [3]. Ona podrazumijeva već razvijene i testirane mrežne sisteme, definisane u Verilog kodu, sa jednostavnim dizajnom, i ima za cilj da omogući jednostavne izmjene i nadogradnju prilikom kreiranja različitih mrežnih uređaja. Međutim, skalabilnost tog dizajna uslovila je ograničenje performansi uzrokovano serijskom obradom Ethernet frejmova. Zbog ove činjenice i sa željom da se dizajnira mrežni uređaj visokih performansi, u ovom radu je kreirana sopstvena arhitektura Ethernet komutatora koja treba da omogući paralelnu obradu frejmova sa svih portova NetFPGA razvojne platforme.

Komutator je realizovan upotrebom *crossbar* komutacione matrice sa baferima u ukrsnim tačkama [4], [5], čime je dat doprinos istraživanju ovakvog dizajna mrežnog uređaja. Naime, dugo je zbog kompleksnosti i tehnoloških ograničenja bilo teško realizovati komutator korišćenjem pomenute arhitekture. To je uslovilo da ona ne bude dovoljno istražena. Međutim, u poslednje vrijeme na tom polju je postignut veliki napredak, zahvaljujući istraživanjima u okviru kojih su analizirane performanse ove arhitekture [6]–[8]. Značaj realizacije jednog ovakvog mrežnog uređaja se ogleda u činjenici da će njegovo korišćenje i nadogradnja dati konkretne (a ne samo simulacione) rezultate po pitanju efikasnosti pomenute arhitekture i performansi koje ona sa sobom donosi. Takođe, predloženi dizajn se odlikuje visokom efikasnošću i mogućnošću paralelne obrade podataka sa svih portova NetFPGA kartice (za razliku od referentnog dizajna). Pri tome, ovako dizajniran komutator je moguće unapređivati i nadograđivati, što otvara mogućnost kreiranja složenijih mrežnih uređaja i istraživanja na polju razvoja novih mrežnih protokola.

Nikola Ljumović, Crnogorski Telekom, Podgorica, Crna Gora (tel.: 38267607670, e-mail: nikola.ljumovic@telekom.me).

Danilo Mišović, Crnogorski Telekom, Podgorica, Crna Gora (e-mail: danilo.misovic@telekom.me).

M. Radonjić, Univerzitet Crne Gore, Elektrotehnički fakultet, Podgorica, Crna Gora (e-mail: m.radonjic@ieee.org).

I. Radusinović, Univerzitet Crne Gore, Elektrotehnički fakultet, Podgorica, Crna Gora (e-mail: igorr@ieee.org).

U drugom poglavlju ovog rada dat je kratak opis NetFPGA razvojne pločice i njenih osnovnih komponenti. Struktura i način funkcionisanja glavnih hardverskih cjelina (modula) predloženog dizajna Ethernet komutatora su prikazani u trećem poglavlju. Na kraju su, u četvrtom poglavlju, date završne konstatacije i zapažanja uz pregled mogućnosti daljeg rada i istraživanja na ovom polju.

II. NETFPGA RAZVOJNA PLATFORMA

NetFPGA predstavlja veoma moćnu razvojnu platformu koja omogućava istraživačima da praktično implementiraju sopstvena hardverska rješenja mrežnih elemenata. NetFPGA projekat je pokrenut usled nedostatka adekvatne hardverske platforme za podučavanje i razvoj računarskih mrežnih sistema. Grupa istraživača sa Stanford Univerziteta u Sjedinjenim Američkim Državama osposobila je, 2001. godine, jednu od prvih hardverskih platformi sa ogromnim potencijalom za primjenu u edukativne i istraživačke svrhe u oblasti računarskih mrežnih sistema [9]. Kao reprogramabilna logička komponenta je izabran FPGA čip, pa je i platforma dobila ime NetFPGA, zbog mogućnosti priključenja na mrežu.

NetFPGA pločica posjeduje Xilinx-ov Virtex-II Pro 50 FPGA čip koji radi na taktu od 125 MHz i na kojem se smješta korisnički definisana logika. Okružuju ga četiri memorijska bloka (dva SRAM i dva DDR2 SDRAM). Pločica posjeduje četiri Ethernet 1000BASE-T primopredajnika koja funkcionišu na fizičkom nivou i zaduženi su za slanje i prijem paketa. Pločica je sa host računarom, preko koga se vrši njeno programiranje, povezana PCI interfejsom.

III. HARDVERSKI DIZAJN

Referentna arhitektura sadrži tri glavne hardverske cjeline. To su:

- **Ethernet I/O blokovi** - skup modula koji se direktno povezuju na primopredajne portove i služe za prijem/slanje Ethernet frejmova fizičkim linkom.
- **CPU DMA blokovi** - skup modula koji služe za komunikaciju između host računara i NetFPGA platforme.
- **Hardverska cjelina za obradu toka podataka - *User Data Path***. Ova cjelina je izdvojena i u njoj se vrši sva funkcionalna obrada Ethernet frejmova. Upravo u okviru ove hardverske cjeline je realizovan kompletan dizajn Ethernet komutatora sa baferima u ukrsnim tačkama. Pri tome, kreirana je u potpunosti nova arhitektura mrežnog uređaja u odnosu na arhitekturu realizovanu u okviru referentnog *User Data Path*-a.

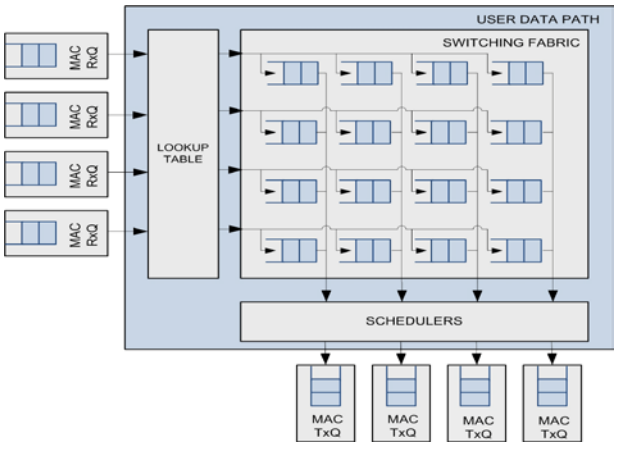
Predloženi dizajn Ethernet komutatora podrazumijeva *crossbar* arhitekturu sa baferima u ukrsnim tačkama (Sl. 1.). Prva faza obrade podataka se obavlja u hardverskim modulima (*MAC RxQ*) u kojima su realizovani prijemni baferi. Ethernet frejmovi u ovu fazu dolaze sa fizičkih portova NetFPGA kartice. Ovi moduli segmentiraju Ethernet frejm na 64-bitne riječi (pakete) i dodaju tzv. *IOQ* 64-bitno zaglavlje u kojem se nalazi informacija o veličini frejma kao i broj porta sa kojeg dolazi frejm. Ovo zaglavlje se dodaje na početak frejma i koristi prilikom desegmentacije, odnosno provjere validnosti frejma.

Podaci se iz ovih modula šalju do *User Data Path*-a. Hardverski dizajn Ethernet komutatora realizovan je u okviru ove hardverske cjeline kroz 3 modula: *Lookup_table, Switching_fabric* i *Schedulers*. Ovi moduli vrše kreiranje i ažuriranje tabele prosleđivanja, pronalaženje izlaznog porta, upisivanje u bafere u ukrsnim tačkama i slanje 64-bitnih riječi do hardverskih modula u okviru kojih se vrši restauracija originalnih frejmova.

Nakon napuštanja *User Data Path*-a frejmovi dolaze do predajnih hardverskih modula (*MAC TxQ*), analognih prijemnim, u okviru kojih su realizovani predajni baferi. Ovi moduli uklanjaju *IOQ* zaglavlje i šalju podatke na fizičke portove NetFPGA kartice.

Međusobna komunikacija prikazanih modula realizovana je korišćenjem 4 signala i to *DATA*, *CTRL*, *WR* i *RDY*. *DATA* je 64-bitni signal i služi za prenos 64-bitnih riječi segmentiranog Ethernet frejma. *CTRL* signal se sastoji od 8 bita i prenosi se istovremeno sa *DATA* signalom. Njegova uloga je da definiše koja se 64-bitna riječ trenutno prenosi *DATA* linijama (da li se radi o *IOQ* zaglavlju, poslednjoj riječi frejma itd.). *RDY* signalom naredni modul signalizira prethodnom da je spreman da primi podatke (ovaj signal je uvijek usmjeren suprotno od preostala 3 signala). Nakon toga, prethodni modul počinje sa slanjem podataka (ukoliko ih ima). *WR* signalom on daje informaciju narednom modulu da je trenutno aktivan prenos na *CTRL* i *DATA* magistralama.

RDY i *WR* signali mogu biti jednobitni ili 4-bitni. U prikazanoj realizaciji komunikacija modula *Lookup_table* sa komutacionom matricom osim *DATA* i *CTRL* signalima vrši se korišćenjem 4-bitnih *WR* signala. Naime, *Lookup_table* šalje komutacionoj matrici 4-bitni *WR* signal koji ne predstavlja samo indikator slanja podataka, već i definiše u koje bafere odgovarajućeg reda komutacione matrice će podaci biti upisani (svaki bit kontroliše upis u jedan od bafera). Modul *Schedulers* sa komutacionom matricom komunicira sa *DATA*, *CTRL* i 4-bitnim *WR* i *RDY* signalima. Ovaj modul, korišćenjem 4-bitnog *RDY* signala, vrši signaliziranje baferima kolone komutacione matrice da mogu poslati paket. Nakon toga, komutaciona matrica korišćenjem 4-bitnog *WR* signala indicira slanje podataka ali i prosleđuje informaciju o tome koji bafer kolone komutacione matrice u tom trenutku vrši slanje. U ostalim slučajevima *WR* i *RDY* signali su jednobitni.



Sl. 1. Hardverski dizajn Ethernet komutatora

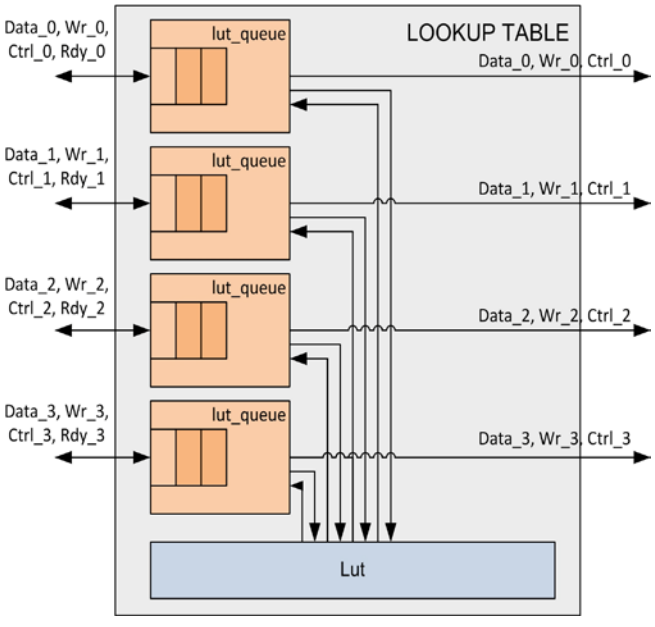
A. *Lookup_table* modul

U okviru *Lookup_table* modula definisana su 4 (*Lut_queue*) podmodula koji odgovaraju ulaznim portovima i podmodul *Lut* u kome je definisana tabela prosleđivanja (Sl. 2.). U podmodulima *Lut_queue* se nalaze baferi za smještanje frejmova dok se pronađe izlazni port na koji treba proslijediti pristigli frejm. U cilju pronalaženja izlaznog porta ovi podmoduli konsultuju tabelu prosleđivanja u podmodulu *Lut*. *Lut* vrši ažuriranje tabele prosleđivanja i interakciju sa njom.

Kada u *Lut_queue* pristigne *IOQ* zaglavlje ovaj podmodul prelazi u stanje čekanja prve 64-bitne riječi Ethernet frejma. Kada ona stigne *Lut_queue* iz nje izvlači destinacionu *MAC* adresu (prvih 48 bita) i prvih 16 bita izvorišne *MAC* adrese. Zatim ovaj podmodul prosleđuje izdvojenu destinacionu *MAC* adresu podmodulu *Lut*. Po pristizanju druge riječi Ethernet frejma *Lut_queue* kompletira izvorišnu *MAC* adresu izvlačenjem preostalih 32 bita i prosleđuje je podmodulu *Lut*. Nakon toga, *Lut_queue* čeka da *Lut* konsultuje (i eventualno ažurira) svoju tabelu prosleđivanja i pronađe destinacioni port na koji treba proslijediti frejm.

Podmodul *Lut* po pristizanju destinacione *MAC* adrese pretražuje da li ta adresa i njoj odgovarajući port postoji u tabeli prosleđivanja. Ukoliko postoji, *Lut* saopštava broj porta na koji treba proslijediti frejm podmodulu *Lut_queue*. Ukoliko ne pronađe tu adresu, podmodulu *Lut_queue* prosleđuje *broadcast* vrijednost na osnovu koje će frejm biti proslijeđen na sve portove osim porta sa kojeg je frejm stigao. Po pristizanju izvorišne *MAC* adrese *Lut* pretražuje tabelu prosleđivanja. Ukoliko utvrdi da ta MAC adresa ne postoji u tabeli prosleđivanja on vrši njeno upisivanje u paru sa odgovarajućim portom.

Nakon što mu *Lut* proslijedi broj izlaznog porta, podmodul *Lut_queue* počinje sa slanjem podataka modulu *Switching_Fabric* podešavajući vrijednost *WR* signala saglasno informaciji o izlaznom portu na koji treba proslijediti frejm.



Sl. 2. *Lookup_table* modul

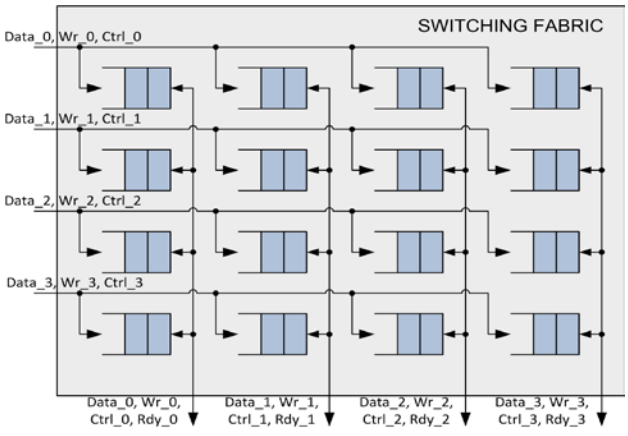
B. *Switching_fabric* modul

Modulom *Switching_fabric* realizuje se komutaciona matrica koju karakterišu četiri ulaza, četiri izlaza i 16 bafera u ukrsnim tačkama (Sl. 3.). Svaki ulaz odgovara jednom redu komutacione matrice i podrazumijeva tri magistrale i to: 64-bitnu *DATA* magistralu, 8-bitnu *CTRL* magistralu i 4-bitnu *WR* magistralu. Svaki od 4 bita *WR* magistrale kontroliše upisivanje podataka u jedan bafer u redu komutacione matrice koja odgovara tom ulaznom portu. Bafer u ukrsnoj tački po pristizanju podataka provjerava da li njemu namijenjen bit ima vrijednost logičke jedinice. Ukoliko ima, upisuje podatke sa *DATA* i *CTRL* magistrala. U suprotnom, podaci se odbacuju.

Izlazi iz ovog modula su realizovani preko *DATA*, *CTRL* i *WR* magistrala. Međutim, komunikacija sa narednim modulom (*Schedulers*) podrazumijeva i korišćenje 4-bitnih *RDY* magistrala kojima on vrši signaliziranje baferima u koloni komutacione matrice da mogu poslati paket. Izlazna *WR* magistrala u ovom slučaju služi da proslijedi informaciju narednom modulu o tome koji bafer u koloni komutacione matrice trenutno vrši slanje podataka.

Switching_fabric je modul koji ne vrši nikakvu obradu paketa. Njegov zadatak je da prihvati paket, smjesti u odgovarajući bafer i isčita iz istog u trenutku kada se steknu uslovi za to. Kao što se može primijetiti, komunikacija *Switching_fabric* i *Lookup_table* modula ne podrazumijevaju postojanje *RDY* magistrala, odnosno *Switching_fabric* ne signalizira kada je spreman da primi podatke. To znači da modul *Lookup_table* uvijek šalje pakete odmah nakon obrade, bez obzira na broj slobodnih mjesta u baferima. U slučaju da je bafer u koji treba upisati paket pun, paket će biti odbačen [5].

Baferi u kojima se nalaze paketi, koje je potrebno proslijediti na isti izlazni port, su povezani na zajedničku izlaznu magistralu. Slanje paketa iz bafera na magistralu je realizivano po *round robin* principu. Za realizaciju *round robin* algoritma odlučivanja odgovoran je *Schedulers* modul. Naime, *Schedulers* kao naredni modul u svakom taktu šalje svoje *RDY* signale (za svaku kolonu matrice po jedan) koji na jednom od svoja četiri bita imaju logičku jedinicu. Svaki bit *RDY* magistrale određenog izlaza (kolone) odgovara jednom baferu u koloni komutacione matrice i služi za signaliziranje tom baferu da može poslati paket. Bafer, ukoliko nije prazan, postavlja podatke na zajedničku magistralu, ali i logičku jedinicu na mjestu koje mu pripada u izlaznom *WR* signalu. Na taj način svaki



Sl. 3. *Switching_fabric* modul

izlaz komutacione matrice svojim *WR* signalom signalizira narednom modulu da na magistrali postoji paket koji bi trebalo prihvatiti i dalje procesuirati.

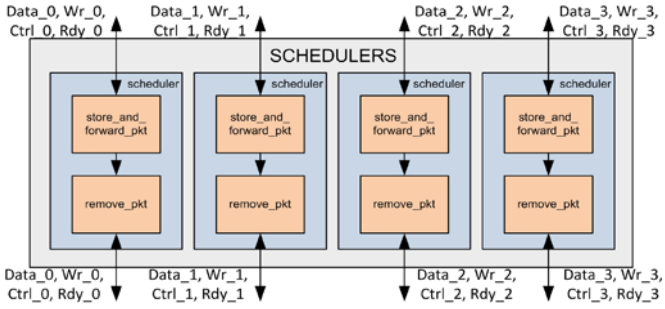
C. *Schedulers* modul

Schedulers je modul koji je zadužen za definisanje algoritma posluživanja bafera komutacione matrice i pri tome obavlja funkciju desegmentacije Ethernet frejma. Ovaj modul se sastoji od 4 podmodula (4 *Scheduler*-a) kao što je prikazano na Sl. 4. Svaki od tih podmodula odgovara jednoj koloni komutacione matrice odnosno jednom izlaznom portu. Oni, pomoću svojih *RDY* magistrala, u svakom taktu vrše signaliziranje različitom baferu u koloni komutacione matrice da može poslati paket.

U okviru svakog *Scheduler*-a realizovana su 2 podmodula i to: *Store_and_forward_pkt* i *Remove_pkt.* S obzirom da se u svakom taktu poslužuju baferi koji odgovaraju različitim ulaznim portovima, paketi koji dolaze u *Scheduler* modul pripadaju različitim Ethernet frejmovima. Upravo zbog toga, *Store_and_forward_pkt* se sastoji od 4 bafera, pri čemu svaki od njih odgovara jednom od bafera u ukrsnim tačkama u odgovarajućoj koloni komutacione matrice. To znači da *Schedulers* modul ima ukupno 16 bafera, odnosno da svaki od bafera u ukrsnim tačkama komutacione matrice ima sebi namijenjen bafer u *Schedulers* modulu, čime se vrši desegmentacija početnog frejma. Koji od bafera *Store_and_forward_pkt*-a će prihvatiti paket određuje *WR* signal. Podaci sa ulazne magistrale dolaze na svaki od ovih bafera ali se upisuju samo u onaj kojem odgovara logička jedinica *WR* signala.

Nakon smještanja frejma, vrši se provjera njegove validnosti posmatranjem broja 64-bitnih riječi u baferu. Ukoliko taj broj odgovara broju definisanom u *IOQ* zaglavlju, frejm je validan. Time se potvrđuje da u procesu obrade nije došlo do gubitka nijednog od segmenata frejma, odnosno da je frejm kompletan. Nekompletni frejmovi se odbacuju.

Remove_pkt podmodul provjerava da li u baferima *Store_and_forward_pkt*-a postoje validni frejmovi za slanje. Ukoliko postoje i ukoliko je *MAC TxQ* modul tog izlaznog porta spreman za prijem, ovaj podmodul ih uklanja iz *Store_and_forward_pkt*-a i prosleđuje *MAC TxQ* modulu. Kako je frejmove iz sva četiri bafera *Store_and_forward_pkt*-a potrebno prosleđivati na isti izlazni port, neophodno je izabrati odgovarajući način posluživanja tih bafera. I u ovom slučaju, kao najjednostavniji, je izabran *round robin* način posluživanja. Nakon prijema frejma *MAC TxQ* uklanja *IOQ* zaglavlje i šalje ga na fizički port NetFPGA kartice.



Sl. 4. *Schedulers* modul

IV. ZAKLJUČAK

U ovom radu predstavljena je hardverska realizacija *crossbar* Ethernet komutatora sa baferima u ukrsnim tačkama na NetFPGA platformi. Kreirana je nova arhitektura mrežnog uređaja koja omogućava brzu i efikasnu obradu i prosleđivanje paketa. Dalji rad na ovoj arhitekturi, njeno detaljno testiranje i nadogradnja, omogućiće poređenje rezultata koji se dobijaju u realnom okruženju sa rezultatima simulacija i analitičkih modela.

Takođe, planirana implementacija dodatnog hardvera i softvera će obezbijediti kvalitetnu obradu statističkih podataka koji se odnose na propusnost komutatora, srednje kašnjenje paketa, varijacije u kašnjenju, izgubljene pakete i sl. Ove analize će predstavljati dobru osnovu za dalja unapređenja predloženog dizajna.

LITERATURA

[1] S. Palnitkar " *Verilog - A Guide To Digital Design And Synthesis*", Prentice Hall, 2003.

[2] B. Urgen, M. Radonjić, I. Radusinović, *"Implementacija generatora paketskog saobraćaja na NetFPGA platformi*", Proc. of *TELFOR 2009*, Serbia, november 2009, pp. 282-285

[3] Stanford University, NetFPGA homepage, Users Guide, http://netfpga.stanford.edu.

[4] J. Cvorovic, I. Radusinovic, M. Radonjic, "*Buffering in Crosspoint-Queued Switch*", Proc. of *TELFOR 2009*, Serbia, november 2009, pp. 198-201.

[5] Y. Kanizo, D. Hay, I. Keslassy, *"The Crosspoint-Queued Switch", Proc of IEEE Infocom '09, Brazil* 2009*, pp. 729-737.*

[6] M. Radonjic, I. Radusinovic, *"Impact of scheduling algorithms on performance of crosspoint-queued switch"*, Annals of telecommunications, to be published.

[7] M. Radonjic, I. Radusinovic, *"Buffer Length Impact to 32x32 Crosspoint Queued Switch Performance"*, Proc. of ISCC 2010, Riccione, Italy, 2010, pp. 954-959.

[8] M. Radonjic, I. Radusinovic, *"Average Latency and Loss Probability Analysis of Crosspoint Queued Crossbar Switches"*, Proc of Elmar-2010, Zadar, 2010, pp. 203-206.

[9] G. Watson, N. McKeown, and M. Casado, "*Netfpga - a tool for network research and education*", In 2nd Workshop on Architecture Research using FPGA Platforms (WARFP), Austin, TX, 2006.

ABSTRACT

In this paper implementation of the Ethernet switch on the NetFPGA platform is presented. Switch is implemented using the crosspoint queued crossbar switching architecture. The described switch effectively processes and forwards Ethernet frames that arrive at its ports at speeds of one gigabit per second. The significance of proposed implementation lies in the possibility of testing and upgrading the aforementioned switch architecture. That will allow the continuation of work carried out so far on the analysis of performance of the CQ switch, by comparing simulation and analytical results with the results obtained in the real network environment.

CQ ETHERNET SWITCH IMPLEMENTATION ON THE NETFPGA PLATFORM

Nikola Ljumović, Danilo Mišović, Milutin Radonjić, Igor Radusinović