Zbornik radova Fakulteta tehničkih nauka, Novi Sad

**UDK: 621.39**
**DOI: https://doi.org/10.24867/24BE06Milojica**

# SIMULACIJA TT SAOBRAĆAJA I ANALIZA PERFORMANSI ČVORA BAZIRANOG NA NAMJENSKOJ FPGA PLATFORMI U ETHERNET MREŽI

# SIMULATION OF TT TRAFFIC AND PERFORMANCE ANALYSIS OF A NODE BASED ON A SPECIFIC FPGA PLATFORM IN THE ETHERNET NETWORK

Dejan Milojica, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – RAČUNARSTVO I AUTOMATIKA**

**Kratak sadržaj –** *U ovom radu su opisane karakteristike Time-Triggered (TT) Ethernet saobraćaja koji se koristi za komunikaciju između jednog master, i vise slejv (eng. slave) kontrolera u mreži. Pristup slejva u mrežu obezbjeđen je kroz interfejs uređaj, koji filtrita primljene pakete, i dostavlja ih do slejv kontrolera. Slejv kontroler predstavlja jedinstvenu Field-programmable gate array (FPGA) platformu sa obezbjeđenim Serial Peripheral Interface (SPI) - TT modulom za komunikaciju sa interfejs kontrolerom. Rad uključuje analizu izvršavanja najzahtjevnijih dijelova koda, kao i uticaj Best-Effort (BE) saobraćaja na prenos TT poruka*

**Ključne reči:** *Time-Triggered Ethernet, FPGA, SPI, master, slejv*

**Abstract** – *This paper describes the characteristics of Time-Triggered (TT) Ethernet traffic, which is used for communication between one master and several slave controllers in the network. The slave's access to the network is provided through an interface device, which filters received packets and delivers them to the slave controller. The slave controller represents an embedded Field-programmable gate array (FPGA) platform with a provided Serial Peripheral Interface (SPI) - TT module for communication with the interface controller. The paper includes an analysis of the execution of the most demanding parts of the code, as well as the impact of Best-Effort (BE) traffic on the transmission of TT messages.*

**Keywords:** *Time-Triggered Ethernet, FPGA, SPI, Master, Slave*

## 1. UVOD

Distribuirani sistemi koji rade u stvarnom vremenu, da bi ostvarili svoje funkcionalnosti, zahtijevaju da se čitav proces upravljanja, prenosa, djelovanja, komunikacije, odvija u stvarnom vremenu, te u idealnom slučaju bez kašnjenja.

Međutim, često to nije ostvarivo korišćenjem već postojećih protokola, kao što su to LIN, CAN, FlexRay i drugih jer su sistemi isuviše kompleksni, sa bezbroj uređaja, senzora, aktuatora i slično.

**NAPOMENA:**
**Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Vladimir Marinković.**

Da bi se obezbjedila komunikacija sa konstantnim i garantovanim kašnjenjem, zahtjeva se upotreba vremenski kritičnih poruka, odnosno upotreba jednog od vremenski kritičnih protokola. Jedan od protokola koji nam to omogućavaju, predstavlja TTEthernet [1].

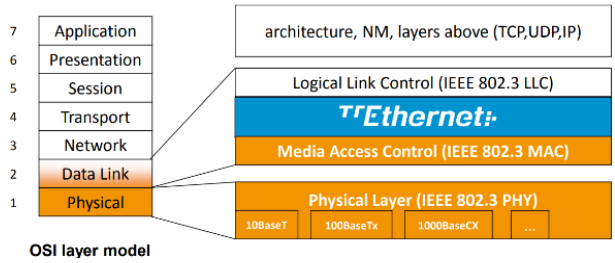
Proširenje Etherneta nastalog prije više od pet decenija, čime se ostvaruje kompatibilnost unazad. Odnosno, ostvaruje se mogućnost prenosa vremenski kritičnih, ali i standardnih ET (eng. *Event Triggered)* poruka, korišćenjem iste fizičke infrastrukture [4]. Bazira se na tome da nam ovaj protokol garantuje maksimalno ostvarivo kašnjenje za vremenski kritične poruke, koje su neophodne za funkcionalnost jednog kompleksnog sistema za rad u stvarnom vremenu. Upotreba TTEtherneta u današnje vrijeme se bazira prvenstveno na transportnoj industriji, gledajući većinom na automobilsku ali i avio kao i svemirsku, dok u zadnje vrijeme svoju upotrebu pronalazi i u ostalim industrijama, gdje imamo sisteme za obnovljive izvore energije, i slično. Cilj istraživanja je da se omogući realizacija simulatora vremenski kritičnog saobraćaja između jednog Master, i jednog ili više slejv uređaja u mreži. Odnosno, ideja se bazira na tome da kreirani simulator može da se iskoristi u realnim situacijama sa adekvatnim hardverom, s tim da je neophodno modifikovati korisne podatke poruka koje se prenose, tako da budu smislene. Takođe, uzeta je u obzir i analiza prenosa ovih poruka, potencijalna "zagušenja" veličinama korisnog sadržaja, brojem slejv čvorova, obradom određenih pristiglih poruka, brojem poruka i sličnih stvari

## 2. KARAKTERISTIKE TTEthernet MREŽE

TTEthernet predstavlja komunikacioni protokol u realnom vremenu (eng. *real-time)* otporan na greške (eng. *fault-tolerant*) koji se koristi u bezbjedonosno - kritičnim sistemima kao što su transportna industrija, industrijska automatizacija. Kompatibilan je sa IEEE 802.3 Ethernetom i adekvatno se integriše sa postojećim komponentama Ethernet mreže, što je pri realizaciji i bio jedan od primarnih uslova. Omogućava determinističu komunikaciju u realnom vremenu kao i prenos asinhronog saobraćaja, paralelno na istoj Ethernet mreži. Odnosno, omogućava prenos vremenski kritičnog saobraćaja, ali i standardnog nekritičnog Ethernet saobraćaja (eng. *non-critical*), korišćenjem zajedničke fizičke infrastrukture, uz striktnu garanciju kašnjenja za TT saobraćaj [3]. Na slici 1, prikazana je struktura *Open Systems Interconnection* (OSI) modela, kao i reprezentacija TTEthernet-a. U

TTEthernet mrežama razlikujemo 4 vrste saobraćaja kao i tipova poruka [1]:

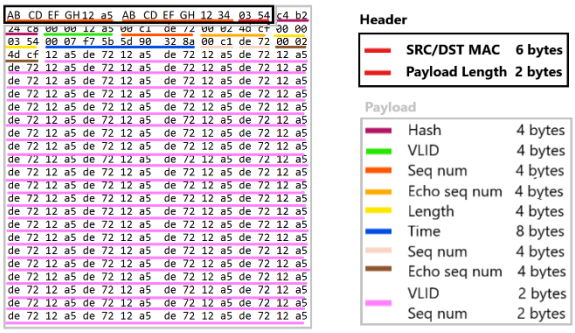
- Deterministički / Vremenski kritični - TT,
- Sporadični (eng. *Rate-Constrained*) - RC,
- Standardni -BE,
- Specijalni kontrolni (eng. *Protocol Control Frames*) – PCF




Slika 1. *TTEthernet kao sastavni dio OSI modela* [2]

## 3. STRUKTURA TTETHERNET PORUKA

Na sljedećoj slici, dat je primjer jednog TTEthernet paketa, zabilježenog u pretposljednjem sloju veze podataka, u realnoj primjeni prenosa između dva uređaja u mreži.




Slika 2. *Struktura Ethernet paketa*

TTEthernet mreža predstavlja „izvedenu“ Ethernet mrežu, kao takva, zadržava karakteristike što se tiče same strukture paketa. U skladu sa tim, TTEthernet paket sastavljen je iz 2 dijela, odgovarajuceg TTEthernet zaglavlja (eng. *Header*) kao i korisnih podataka (eng. *Payload*).

Zaglavlje čine po 6 bajtova izvorišne i odredišne MAC (eng. *Medium Access Control*) adrese, kao i 2 bajta dužine korisnih podataka. Korisni podaci predstavljaju korisnički sadržaj koji ima za cilj da prenese informacije od interesa. Za potrebe rada simulatora TT saobraćaja, definisana je probna stuktura korisnih podataka, u skladu sa potrebama analize i obrade pristiglih paketa.

## 4. STRUKTURA TTETHERNET MREŽE

### 4.1. Master čvor

Predstavlja master kontroler u mreži koji ima za zadatak da upravlja radom, i prati status slejv uređaja u mreži, koji zajedno definišu jedan sistem. Odgovarajući sistem, može/mora da ima jedan Master uređaj, kao i jedan ili više slejv uređaja. Radi se o računaru sa custom Linux operativnim sistemom koji sadrži PCIe mrežnu karticu za koju je razvijen Linux kernel drajver pomoću kojeg se vrši komunikacija.

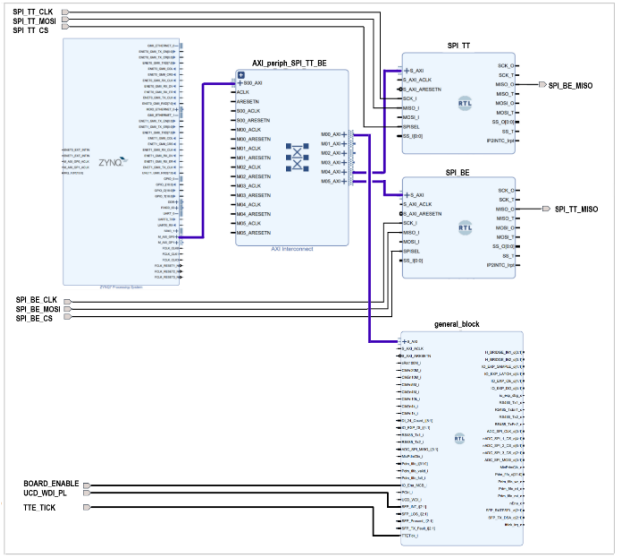
### 4.2. Slejv čvor

Slejv uređaj predstavlja sistem na ploči (eng. *System-on-Chip - SoC*) realizovan kao jedinstvena FPGA ugrađena računarska platforma, koja se koristi za TT komunikaciju u TTEthernet mrežama. Pojam “jedinstvena FPGA platforma”, navedena je iz razloga što se dati čvor za komunikaciju može koristiti na više mogućih načina, što je omogućeno korišćenjem kompleksne hardverske strukture. Dodatno, “izbor” načina funkcionalnosti obezbjeđen je kroz različite FPGA blokove, koji preko AXI (eng. *Advanced eXtensible Interface*) periferala obezbjeđuju komunikaciju sa *Zynq* procesorom [5]. Jedan način upotrebe Slejv čvora, bazira se na tome da ga koristimo za komunikaciju bez “prisustva” *End System*-a, što nas dovodi do toga da se on ne može samostalno posmatrati kao mrežni uređaj, odnosno ne može samostalno da učestvuje u mrežnoj komunikaciji. Zahtijeva se prisustvo dodatnog uređaja, koji će proširiti slejv, “premostiti” do mreže, te ga nazivamo interfejs čvor. Imamo dva nezavisna bloka SPI TT kao i SPI BE, koji se koriste za TT odnosno ET saobraćaj sukcesivno. Ideja je takva da svaki od njih preko zajedničkog AXI periferala, obezbjeđuje komunikaciju sa procesorom. Takođe, ideja je da razdvajanjem vremenski kritičnog i nekritičnog saobraćaja, povećamo performance izvršavanja, ali i na neki način pojednostavimo prenos različitih poruka. Već je navedeno da u ovom stanju, Slejv nije u mogućnosti da direktno pristupa mreži, već se pristup obezbjeđuje kroz interfejs uređaj. Prenos poruka ka datom uređaju, iz SPI TT i SPI BE blokova, odvija se preko SPI-a. Kao što se vidi na slici, ukoliko se poruka šalje sa Slejv-a, tada se podaci šalju na **SPI_BE_MISO**, odnosno **SPI_TT_MISO,** dok ukoliko poruka pristiže ka Slejv uređaju, tada se koriste **SPI_TT_MOSI**, odnosno **SPI_BE_MOSI**, u zavisnosti od toga o kom tipu poruke govorimo. U skladu sa tim, poruka se pri prenosu prebaciva u odgovarajući SPI bafer periferala, otkud se prenosi u odgovarajućem smjeru. Za SPI_TT i SPI_BE, možemo reći da su oni AXI u SPI, odnosno SPI u AXI konvertori. U zavisnosti od toga da li Slejv šalje podatke, ili prihvata, dati podaci se preko SPI-a prenose do odgovarajućeg bloka, te dalje preko AXI-a, ili obrnuto.

### 4.3. Interfejs uređaj

Kako Slejv uređaj ne pristupa direktno mreži, kao takav zapravo nije niti “svjestan” postojanja date mreže. Jedino čega jeste svjestan, je postojanja interfejs uređaja sa kojim komunicira. Kako je interfejs uređaj zapravo ES uređaj, obezbjeđena je potrebna sinhronizacija, odnosno ispunjen uslov za prenos vremenski kritičnih poruka. Međutim, i Slejv uređaj učestvuje u prenosu vremenski kritičnih poruka, ali kako nije ES, ali ni SW uređaj, nije svjestan pojma sinhronizacije, kao ni samog vremena. To je zapravo osnovna uloga interfejs uređaja, da “pomogne” Slejv-u da svoje poruke korektno i pravovremeno pošalje u mrežu. Za interfejs uređaj se može reći da ima niz senzore za mjerenje, od kojih su najznačajniji senzor za temperaturu, žiroskop kao i akcelerometar. Vrijednosti njihovog mjerenja se mogu prenositi korišćenjem *peer* paketa, ili TT paketa u zavisnosti od toga šta treba da bude odredište, odnosno na šta će te vrijednosti uticati. *Peer* poruke predstavljaju specijalnu vrstu interfejs – Slejv poruka, čiji primarni cilj predstavlja prenos signala

od realnih interesa, ka Slejv. Kao što je objašnjeno, interfejs uređaj ima više integrisanih senzora za mjerenje, čije signale zapravo treba prenijeti.




Slika 3. *SPI TT i SPI BE FPGA blokovi*

## 5. DIJAGRAM TOKA IZVRŠAVANJA SIMULATORA

Na slici 4 prikazan je dijagram toka, koji prati tok izvršavanja simulatora. Izvršavanje se obavlja u jednoj programskoj niti. Tok izvršavanja se može podijeliti u nekoliko faza.

### 5.1. Inicijalizacija

Inicijalizacija ima za zadatak da pripremi sistem za potpunu funkcionalnost. U skladu sa tim, neophodno je učitati fajlove koji definišu raspored prenosa poruka, tačno definisana vremena prenosa svake od njih, period prenosa, kao i SPI konfiguracioni fajl, koji nam omogućava pravilno konfigurisanje SPI-a.

### 5.2. Provjera postojanja sinhronizacionog takta

Nekoliko puta je pojašnjena uloga slejv uređaja, odnosno pristupačnost interfejs uređaja koji obezbjeđuje sinhronizacioni takt ka slejv-u. Pristup datom taktu ostvaruje se kroz virtuelni sistem datoteka *sysfs*, tako da je iz aplikacije neophodno konstantno čitanje datih vrijednosti globalnog časovnika, usporedba sa proteklim vremenom lokalnog časovnika, čime ostvaruje spoznaju o početku jednog vremenskog intervala.

### 5.3. Provjera sinhronizacije kroz prihvatanje *peer* poruke

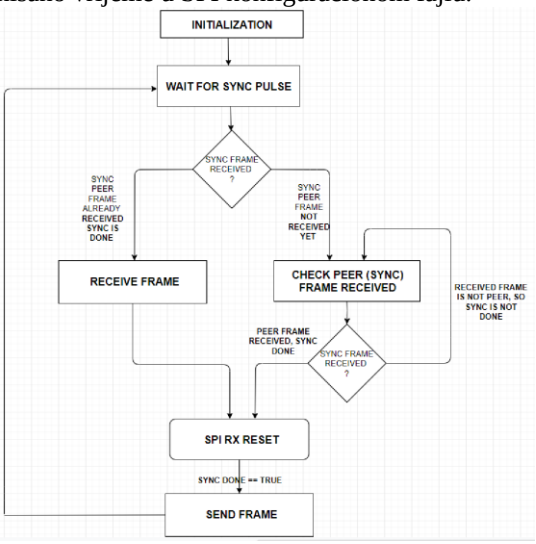
Iako je aplikacija postala svjestna postojanja sinhronizacionog takta koji označava početak vremenskog intervala, problem koji i dalje egzistira je nepoznavanje rednog broja vremenskog intervala. Već od prije je poznato da je veličina jednog vremenskog intervala 1ms, odnosno da je trajanje jednog perioda 10ms (definisano konfiguracionim fajlovim sa početka), odnosno, u jednom periodu imamo 10 jednakih vremenskih intervala, koji se identifikuju rednim brojevima od 0 do 9. Kako bismo tačno utvrdili po započinjanju rada aplikacije, u kom vremenskom intervalu se nalazimo, neophodno je da iskoristimo dolazne poruke, koji imaju tačno definisan vremenski interval pristizanja. Za tu namjenu, iskorišćeni su *peer* paketi, koji prema rasporedu virtuelnih veza, dolaze periodično u definisanom slotu. Odnosno, ideja je da čekamo na *peer*, čime možemo da ozvaničimo da se radi o „konkretnom“ vremenskom intervalu, te započnemu normalnu funkcionalnost sistema.

### 5.4. Prihvatanje poruka

Po završetku uspješno obavljene sinhronizacije, i kada je aplikacija postala svjesna rednog broja vremenskog intervala, stvarna funkcionalnost može da počne. U ovoj fazi, aplikacija ima za zadatak da prihvata pristigle poruke, odnosno da ih parsira I provjerava da li je došlo do greške prilikom prenosa. Kao što je pomenuto, očekuje se za odgovarajući vremenski interval da se prihvate određeni paketi. Ukoliko u datom intervalu, nema niti jednog paketa za prihvatanje, osnovna sumnja se upućuje na to da smo otklizali sa vremenskim intervalom, i da imamo pogrešnu vrijednost. Na taj način ćemo u sljedećem intervalu ponovo čekati na *peer* poruku, kako bismo potvrdili da se radi o korektnoj vrijednosti.

### 5.5. Slanje poruka

Kako dijagram toka nalaže, nakon što se obavi potreban posao vezan za prihvatanje poruka, neophodno je da pošaljemo poruke od interesa. Prije svega, slanje poruka ne može da počne prije definisanog perioda vremena, u skladu sa konfiguracionim SPI fajlom. Takođe, da bismo izvršili slanje poruka, prije samog početka je neophodno provjeriti da li je SPI bafer za smještanje u potpunosti slobodan. Neophodno je da vrijeme izvršavanja procesa slanja svih poruka, ne bude duže u odnosu na unaprijed definisano vrijeme u SPI konfiguracionom fajlu.




Slika 4. *Dijagram toka izvršavanja simulatora*

## 6. TEORIJSKA OGRANIČENJA

### 6.1. Čekanje na *peer* poruke za ostvarivanje početne sinhronizacije

Osnovni nedostatak faze sinhronizacije je taj što ona u potpunosti zavisi od *peer* poruka sa interfejs uređaja. Kao što je navedeno, ukoliko se dati paket javlja u 9. vremenskom slotu, a mi smo se nalazili u korektnom prvom slotu po pokretanju aplikacije, izgubićemo čitav jedan period komunikacije prije uspostavljanja stabilnog stanja.

### 6.2. Maksimalno vrijeme izvršavanja

Čak i ako programski kod koji se izvršava po prijemu, ali i slanju svake TT poruka prati ograničenja opisana u prethodnom dijelu, teorijski se može desiti da kombinovana vremena izvršenja svega što se izvršava tokom jednog ciklusa integracije, premašuju period integracije. Ova granica se može izraziti jednačinom. (1), gdje je $num\_rcv_{tt+peer}$ broj primljenih poruka, $WCET_{rcv}$ najgori slučaj vremena izvršavanja jednog prijema poruke, $num\_snd_{tt+peer}$ broj poruka za slanje, te $WCET_{send}$ najgore vrijeme izvršavanja jednog slanja poruke. $WCET_{REST,}$ definiše najgore vrijeme izvršavanja ostalih dijelova koda između slanja i prihvatanja.

$$\begin{aligned} &\max\_period \\ &> WCET_{REST} + num_{rcv_{tt+peer}} \cdot WCET_{rcv} \\ &+ num_{snd_{tt+peer}} \\ &\cdot WCET_{send} \end{aligned} \quad (1)$$

## 7. ANALIZA PERFORMANSI SLEJV ČVORA

### 7.1. Analiza izvršavanja pojedinih faza implementacije

U toku izvršavanja aplikacije, neophodno je da budu ispunjeni uslovi po pitanju vremena izvršavanja odgovarajuće faze. Idealan slučaj je da se jedan ciklus koraka završi za jedan vremenski interval, čime bismo u drugom vremenskom intervalu imali „čistu situaciju“ da nastavimo sa istim fazama, ali drugim porukama za prenos. Takođe, česta pojava je ta da kada se pravi raspored, za sve poruke se može pronaći najmanji sadržilac kao vrijednost perioda ponavljanja istih poruka, u odnosu na globalni period rada aplikacije, tako da vremena izvršavanja više nisu ograničena jednim vremenskim intervalom, već najmanjim zajedničkim periodom svih poruka, ali pod uslovom da nemamo nikakvih *offset*-a karakterističnih za prenos, jer u tom slučaju moramo i njih uzeti u obzir. Za jednostavan primjer sa slike 7.1, jedan proces treba da bude završen do početka naredne planirane akcije. Implementacija, se bazira na tome da, od početka vremenskog intervala, 35 – 40% vremena se odvaja da bi SPI postao spreman za rad, kao i da bi odgovarajuće poruke pristigle ka uređaju, zatim se za izvršavanje prihvatanja svih poruka odvaja 25%, koliko se zapravo maksimalno odvaja i za slanje svih poruka. Preostalih 10 – 15% vremena se koristi za restartovanje SPI-a za prihvatanje, odnosno dodatni posao koji se izvršava u niti aplikacije (Provjera postojanja sinhronizacionog takta, čitanje, parsiranje virtuelnog fajla itd.).

### 7.2. Analiza uticaja BE saobraćaja na prenos TT poruka

U skladu sa hardverskom strukturom prikazanom na slici 3, postoje dva odvojena funkcionalna bloka, od kojih je jedan za BE saobraćaj, odnosno drugi za TT. Ideja ovog razvdajanja se bazira na tome da ne smije doći do pojave da BE saobraćaj na bilo koji način naruši prenos TT poruka, jer to može da izazove ogromne posljedice. U skladu sa tim, prenos TT poruka ne zavisi od BE poruka, jer je njihov prenos u potpunosti nezavisan.

## 8. ZAKLJUČAK

Da bi se testirao prenos vremenski kritičnih poruka korišćenjem TTEthernet protokola, neophodno je definisati strukturu uređaja koji formiraju mrežu, definisati njihove mogućnosti, uloge i slično. Nakon fizičke organizacije, i spremnosti uređaja da komuniciraju, iniciranje same komunikacije se obavlja na najvišem sloju, a to je aplikativni sloj. Dakle, kreira se aplikacija koja će imati zadatak da generiše poruke, odnosno prihvata odgovarajuće poruke, u skladu sa unaprijed definisanim vremenskim rasporedom prenosa. Za datu aplikaciju, možemo reći da se zapravo radi o simulatoru prenosa vremenski kritičnog saobraćaja, na poznatoj i jedinstvenoj platformi za izvršavanje. Implementacija simulatora, bazirana je programskoj niti za rad u stvarnom vremenu, čime se daju određene garancije prilikom izvršavanja, odnosno promjene konteksta unutar procesora, jer u suprotnom se može desiti da neki drugi proces, preuzme potpunu kontrolu izvršavanja od strane procesora, i na taj način naruši rad simulatora, a samim tim potencijalno ugrozi ljudske živote.

## 9. LITERATURA


[1] TTTech Computertechnik AG. 2009, “*TTEthernet – A Powerful Network Solution for All Purposes*” "TTEthernet – A Powerful Network Solution for All Purposes" (PDF), posjećeno: 18.07.2022.godine

[2] Jean-Baptiste Chaudron, “*TTEthernet. Theory, Concepts and Applications*”, http://etr2015.irisa.fr/images/presentations/TTEthernet_ETR_2015_Rennes.pdf (PDF), posjećeno: 18.07.2022.godine

[3] Stefan Poledna, Herman Kopetz, Wilfried Steiner, “*Deterministic system design with Time-Triggered technology*”, Microelectronic Systems Symposium (MESS)

[4] Ekarin Suethanuwong, “*Scheduling time-triggered traffic in TTEthernet systems*”, Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference, 17-21 Sept. 2012, Krakow

[5] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, Robert W. Stewart, “*The Zynq Book*”, http://www.zynqbook.com/, posjećeno: 18.07.2022.godine


### Kratka biografija:


**Dejan Milojica** rođen je 1997. godine u Prijedoru, Republika Srpska, Bosna i Hercegovina. Zvanje dipl. Inž. El. stekao na Elektrotehničkom fakultetu u Banja Luci. Godine 2021. upisuje se na master akademske studije na Fakultetu tehničkih nauka u Novom Sadu, odsijek Računarstvo i Automatika. Zaposlen kao Softver Inženjer na institutu RT-RK u Banja Luci.

Kontakt: dejan.milojica@rt-rk