LXIX Konferencija ETRAN, Čačak, 9 - 12. Jun 2025.

# Sistem za prijem podataka sa FMCW radarskog senzora preko LVDS interfejsa

1st Nikola Ž. Petrović
*Katedra za Elektroniku*
*Elektrotehnički fakultet, Univerzitet u Beogradu*
Beograd, Srbija
p.z.nikola@etf.bg.ac.rs

2nd Dušan N. Grujić
*Katedra za Elektroniku*
*Elektrotehnički fakultet, Univerzitet u Beogradu*
Beograd, Srbija
dusan.grujic@etf.bg.ac.rs

***Sažetak*—U ovom radu je prikazan sistem za prijem podataka sa FMCW radarskog sensora pomoću LVDS interfejsa, realizovan na Digilent Nexys Video FPGA platformi. Predloženi sistem omogućа deserijalizaciju podataka primljenih preko LVDS interfejsa, sinhronizaciju podataka, automatsku detekciju širine radarske reči kao i pristiglog CRC podatka. Projektovani sistem vrši CRC proveru kao i adekvatno formatiranje podataka kako bi se omogućila dalja obrada FMCW radarskih podataka.**

***Ključne reči*—Chisel, FPGA, FMCW, LVDS, Radar**

## I. Uvod

U cilju povećanja sigurnosti vozača kao i ostalih učesnika u saobraćaju, sve veći broj proizvođača automobila koristi neki vid naprednih sistema za pomoć pri vožnji (eng. *Advanced Driving Assistance System* – ADAS) [1], [2]. Pri čemu, ovi sistemi često koriste frekvencijski modulisane neprekidno zračeće (eng. *Frequency Modulated Continuous Wave* – FMCW) radare kao primarne senzore usled njihove osobine da funkcionišu pouzdano u svim vremenskim uslovima pri čemu mogu detektovati prepreke i kroz neprozirne materijale. Pored ovoga, FMCW radari su cenovno pristupačni što omogućava njihovu široku primenu.

Da bi se obradili FMCW radarski podaci potrebno je rešiti jedan od ključnih izazova u realizaciji sistema za digitalnu obradu FMCW radarskih podataka. Naime, moderni radarski sistemi generišu veliku količinu sirovih radarskih podataka koju je potrebno obraditi. Ova obrada mora biti u realnom vremenu i sa minimalnim kašnjenjem. Međutim, serijske podatke sa različitih radarskog senzora je potrebno deserijalizovati i formatirati tako da odgovaraju formatu ulaznih podataka DSP akceleratora.

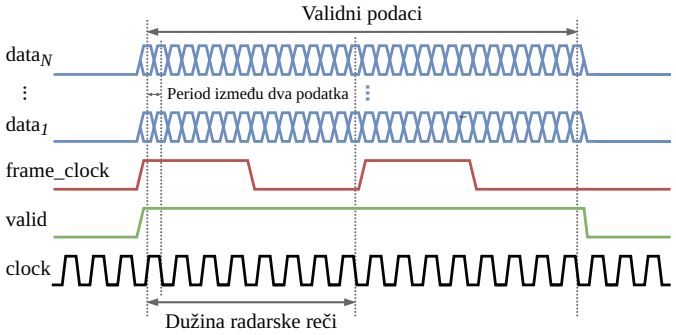
U ovom radu je prikazan sistem za akviziciju podataka sa FMCW radarskog senzora pomoću LVDS (eng. *Low-Voltage Differential Signaling*) interfejsa [3]. Predloženi sistem je projektovan tako da prihvati serijske podatke sa radarskog senzora i formatira podatke tako da su oni spremni za dalju obradu.

Rad je organizovan na sledeći način: u poglavlju II je prikazan princip slanja podataka preko LVDS interfejsa, u poglavlju III je opisana arhitektura sistema kao i informacije o FPGA implementaciji. U poglavlju IV je prikazan zaključak rada.

This work was financially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under contract number: 451-03-137/2025-03/200103.

## II. Opis serijskog protokola

Vremenski dijagram signala AWR2243 [4] radarskog senzora koji se šalju preko LVDS interfejsa u slučaju da se podaci šalju na obe ivice taktnog signala (eng. *Double Data Rate* – DDR), dat je na slici 1.




Slika 1. Ilustracija slanja podataka preko LVDS interfejsa.

LVDS interfejs se sastoji signala za sinhronizaciju podataka označenih sa `valid` i `frame_clock`, taktnog signala `clock` i $N$ diferencijalnih linija preko kojih se šalju podaci označeni na slici 1 sa `data`$_i$ gde $i$ može imati vrednost od 1 do $N$, pri čemu $N$ predstavlja broj kanala.

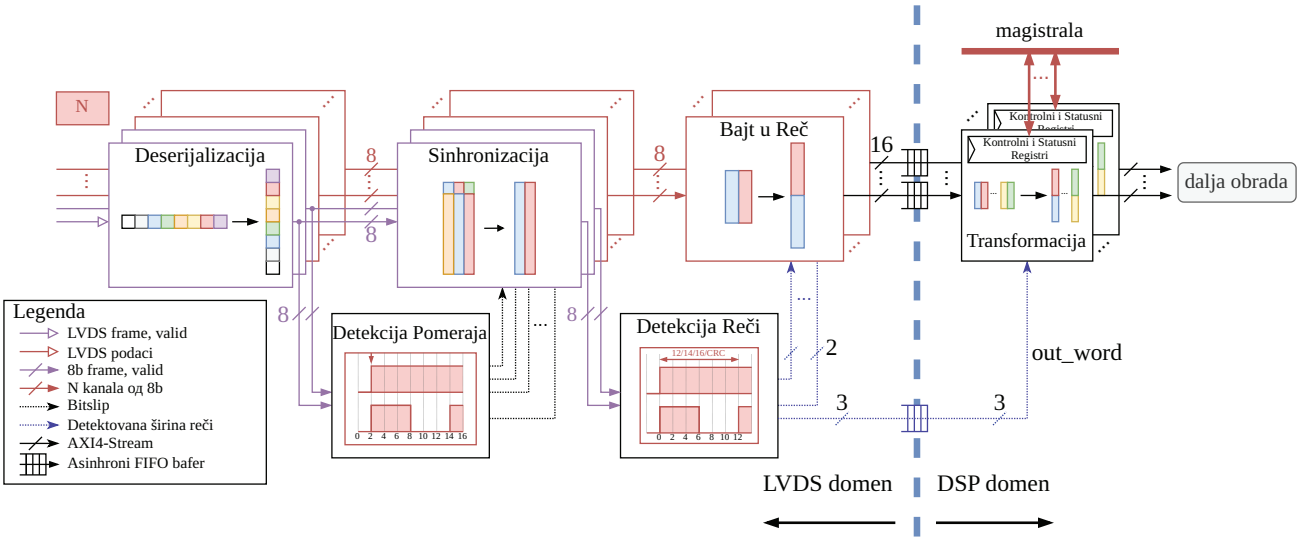
Podaci koji se šalju preko LVDS linija su validni samo dok signal `valid` ima vrednost 1. Sa druge strane, perioda signala `frame_clock` predstavlja trajanje reči koja se šalje. na primer, ukoliko `frame_clock` traje 8 perioda `clock` taktnog signala, i ako se podaci šalju na obe ivice taktnog signala, dužina jedne reči je 16 bita. Pomoću signala `valid` i `frame_clock` moguće je izvući informaciju o širini reči koja se šalje preko LVDS linija.

## III. Opis projektovanog sistema

Dijagram predloženog bloka za prijem i formatiranje podataka prikazan je na slici 2. Sam sistem je napisan u Chisel [5] jeziku za opis hardvera u vidu generatora hardverskih instanci. Ključni parametar predloženog generatora je broj kanala podataka koje je moguće primiti. Ukoliko deo za DSP

ISBN 978-86-6200-032-3 ©2025 AM & ETRAN

https://doi.org/10.69994/69E25018




Slika 2. Blok dijagram predloženog sistema za prijem i formatiranje podataka. $N$ predstavlja broj LVDS podataka koji je moguće primiti.

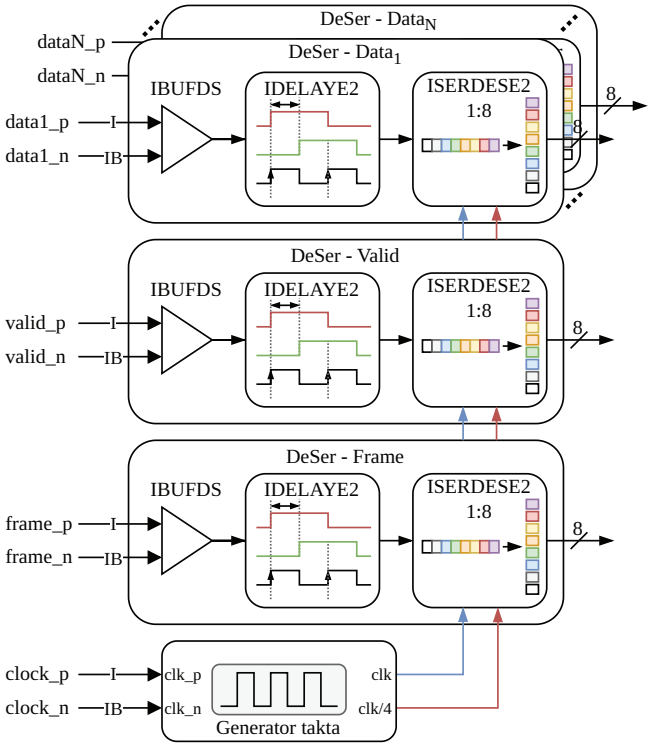
obradu signala radi na različitoj učestanosti od LVDS taktnog signala, moguće je dodati AXI-Stream asinhrone FIFO bafere proizvoljne dubine kao što je to prikazano na slici 2.

### A. *Deserijalizacija podataka*

Prvi korak u prijemu LVDS signala je deserijalizacija podataka. Blok za deserijalizaciju podataka koji ima za cilj da konvertuje primljene serijske podatke u podatke širine jednog bajta je prikazan na slici 3.

Ovaj blok se sastoji od $2 + N$ identičnih jedinica. Prve dve jedinice su zadužene za prijem signala za sinhronizaciju (`valid` i `frame_clock`) dok je ostalih $N$ jedinica namenjeno prijemu podataka. Svaka jedinica se sastoji od bafera koji prima diferencijalni signal i konvertuje ga u jednostrani signal. Kako je za realizaciju sistema korišćena AMD FPGA razvojna ploča, za bafer je korišćen IBUFDS [6]. Kako primljeni LVDS podaci mogu imaju različito kašnjenje, nakon konverzije diferencijalnog signala u jednostrani, potrebno je obezbediti da svaki od primljenih podataka ima isto kašnjenje u odnosu na referentni taktni signal. Da bi se ovo ostvarilo, nakon bafera dodat je blok koji može uneti dodatno kašnjenje signala pod nazivom IDELAYE2 [6].

Na kraju, nakon što su kašnjenja svih signala ujednačena, nalazi se blok za konvertovanje serijskih podataka u podatke širine 8 bita koji je na slici 3 označen sa ISERDESE2 1:8 [6]. Da bi ovaj blok ispravno radio potrebno mu je proslediti dva signala takta. Prvi taktni signal je signal takta serijskih podataka dok je drugi signal takta četiri puta sporiji. Generator takta (sa slike 3) može biti realizovan na jedan od dva načina koji su prikazani na slici 4. U prvom slučaju, instancirana je fazno zatvorena petlja (PLL) i signal koji označava da je PLL u fazi se koristi kao deo kola za reset bloka za prijem podataka. U drugom slučaju, umesto PLL-a se koristi instanca BUFR [6] koja na svom izlazu daje vrednost prosleđenog signala takta kao i njegovu vrednost podeljenu sa četiri. U ovom slučaju

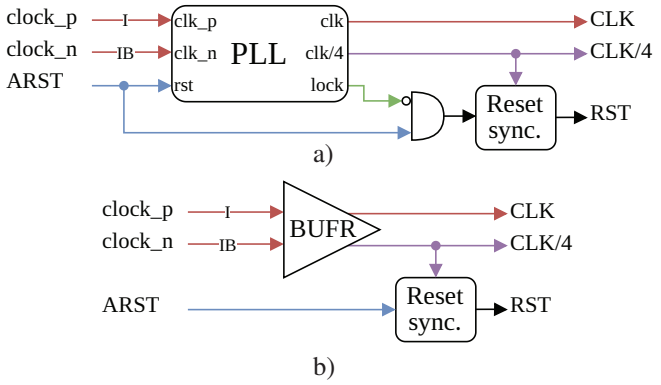



Slika 3. Uprošćeni blok dijagram kola za deserijalizaciju podataka.

asinhroni reset se samo dovodi na kolo za sinhronizaciju. U oba slučaja, reset je sinhronizovan na sporiji signal takta `CLK/4`.

### B. *Sinhronizacija podataka*

Nakon deserijalizacije podataka, potrebno je sinhronizovati podatke s obzirom na to da nije moguće unapred znati poziciju prvog bita pristiglog podatka. Samim tim, u zavisnosti od

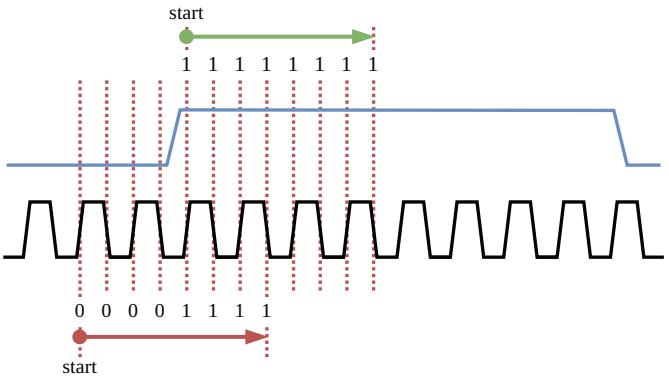



Slika 4. Dva načina za generisanje signala takta i reseta potrebnih za prijem LVDS podataka. U slučaju (a) je instancirana fazno zatvorena petlja koji generiše signal u fazi sa ulaznim signalom takta kao i četiri puta sporiji takt. Signal `lock` koji označava da je fazno zatvorena petlja u fazi se koristi kao deo reset kola. U slučaju (b) za generisanje `CLK` i `CLK/4` se koristi BUFR bafer.

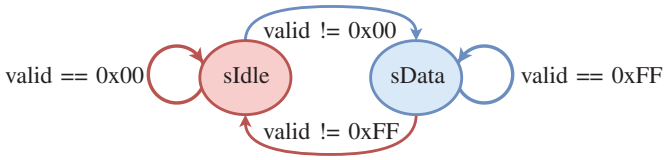
ove pozicije, deserijalizovani podatak može imate različite vrednost. Ovaj problem je ilustrovan na slici 5. U zavisnosti od toga kada je prvi bit očitan, u datom primeru sa slike 5, izlazni podatak bloka ISERDESE2 1:8 imaće vrednost `00001111` (crvena strelica) ili vrednost `11111111` (zelena strelica).

Kako bi se izlazni podaci bloka ISERDESE2 1:8 grupisali u validne podatke, potrebno ih je sinhronizovati. Blok koji određuje za koliko bita je potrebno pomeriti podatke je blok *Detekcija Pomeraja* sa slike 2. Jedan od načina za određivanje ovog broja bita je taj da se posmatra promena `valid` signala sa `0x00` na neku drugu vrednost. Deserijalizovani `valid` podatak može imati samo dve vrednosti, sve nule ili sve jedinice. Ova informacija omogućava da se na osnovu deserijalizovane vrednosti `valid` podatka ustanoviti za koliko je potrebno pomeriti podatke tako da se poravnjaju na odgovarajući način.

Na slici 6 je dat prikaz uprošćene mašine stanja (eng. *Finite State Machine* – FSM) bloka *Detekcija Pomeraja* koji služi za detekciju broja bita za koje je potrebno pomeriti očitane podatke. Mašina stanja ostaje u stanju `sIdle` sve dok se ne detektuje podatak koji nije jednak `0x00`. Zatim,




Slika 5. Primer deserijalizacije podataka u zavisnost od toga gde se nalazi prvi očitani bit.



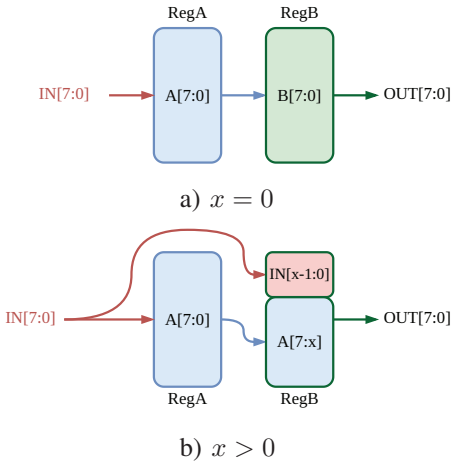

Slika 6. Prikaz uprošćene mašine stanja bloka za detekciju broja bita potrebnih za sinhronizaciju podataka.

u zavisnosti od pročitanog podatka, određuje se vrednost za koju je potrebno pomeriti podatke i mašina stanja prelazi u stanje `sData`. Veza između pročitanih vrednosti i broj bita za koje je potrebno pomeriti podatke `Pomeraj` je data u tabeli I. U okviru stanja `sData`, detektovana vrednost `Pomeraj` se zadržava i mašina stanja ostaje u stanju `sData` sve dok vrednost signala `valid` jednaka `0xFF`. Kada `valid` nije jednak `0xFF`, prelazi se nazad u stanje `sIdle` i ovo označava da je transakcija završena.

Tabela I
VEZA IZMEĐU DESERIJALIZOVANE VREDNOSTI `VALID` SIGNALA I BROJA BITA ZA KOJE JE POTREBNO POMERITI PODATKE.

| valid(bin) | valid(hex) | Pomeraj |
|---|---|---|
| 1111 1111 | 0xFF | 0 |
| 1111 1110 | 0xFE | 1 |
| 1111 1100 | 0xFC | 2 |
| 1111 1000 | 0xF8 | 3 |
| 1111 0000 | 0xF0 | 4 |
| 1110 0000 | 0xE0 | 5 |
| 1100 0000 | 0xC0 | 6 |
| 1000 0000 | 0x80 | 7 |

Nakon određivanja vrednosti `Pomeraj` za koju je potrebno pomeriti podatke, ovu vrednost je potrebno dovesti na blok *Sinhronizacija* kao što je to prikazano na slici 2. Blok *Sinhronizacija* je zadužen za poravnanje podataka na osnovu vrednosti `Pomeraj`. Princip rada ovog bloka prikazan je na slici 7.




Slika 7. Princip rada bloka za sinhronizaciju podataka. Pod a) je dat slučaj kada je vrednost `Pomeraj` jednaka nuli, dok je pod b) dat slučaj kada je vrednost `Pomeraj` različita od nule. Sa $x$ je označena vrednost `Pomeraj`.

Blok za korekciju pomeraja se sastoji od dva registra: RegA i RegB. Podatak širine 8 bita dovodi se na ulaz na registra RegA dok registar RegB predstavlja izlaz bloka. U slučaju kada je vrednost Pomeraj (na slici 7 označeno sa $x$) jednaka nuli, registru RegB se prosleđuje vrednost registra RegA. U suprotnom, kada je Pomeraj različit od nule, potrebno je izvršiti poravnjanje podataka kao što je to prikazano na slici 7 b).

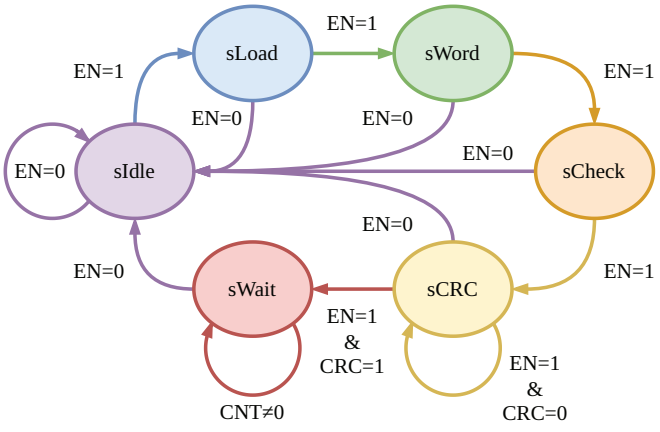
### C. Određivanje širine reči

Nakon sinhronizacije podataka širine 8 bita, potrebno je primljene podatke spakovati u reči odgovarajuće širine. Blok koji određuje širinu reči dat je na slici 2 pod nazivom *Detekcija Reči*. Ovaj blok detektuje širinu reči na osnovu deserijalizovane vrednosti signala frame_clock čija jedna perioda ima trajanje reči koja se šalje [3]. Najčešće širine reči koji radarski senzori podržavaju su 12, 14 i 16 bita za podatke i 32 bita za CRC (ciklična redundantna suma) pri čemu se CRC nalazi na kraju transakcije [4]. Uprošćena mašina stanja ovog bloka je data na slici 8.

Po resetu sistema, mašina stanja ulazi u stanje sIdle i ostaje u ovom stanju dokle god se ne pojavi prvi validni frame_clock podatak. Prvi validni frame_clock podatak se učitava u donjih 8 bita registra regData širine 32 bita i prelazi se u stanje sLoad. Pri čemu je podatak frame_clock validan kada valid ima vrednost 0xFF. Na slici 8, valid je obeležen sa EN dok je vrednost 0xFF predstavljena logičkom jedinicom.

Iz stanja sLoad se prelazi u sledeće stanje sWord i nova vrednost frame_clock se upisuje na mesto drugog bajta registra regData. Kako je minimalna širina podatka koju očekujemo 12 bita [4], situacija da podaci nisu validni u ovom stanju ne bi smela nikada da se dogodi.

Ukoliko su podaci validni (EN=1), u stanju sWord se na osnovu donjih 16 bita registra regData[15:0] određuje da li su podaci širine 12, 14 ili 16 bita. Nova vrednost frame_clock se upisuje u registar regData[23:16] i prelazi se u stanje sCheck. U suprotnom, ukoliko podaci nisu validni, prelazi se u stanje sIdle i ovime je transakcija završena. Tabela II prikazuje detektovane vrednosti širine podataka na osnovu pročitane vrednosti registra regData.




Slika 8. Uprošćena konačna mašina stanja bloka *Detekcija Reči* za određivanje širine reči primljenih podataka.

Tabela II
DETEKTOVANA ŠIRINA REČI U ZAVISNOSTI OD VREDNOSTI REGISTRA REGDATA.

| regData[15:0](hex) | Detektovana širina reči |
|---|---|
| 0x00FF | 16 bita |
| 0xF03F | 12 bita |
| 0xC07F | 14 bita |

U stanju sCheck, očitana vrednost frame_clock podatka smešta se u gornji bajt registra regData[31:24] i prelazi se u naredno stanje sCRC.

U stanju sCRC se na osnovu registra regData i detektovane veličine reči, vrši provera da li je pristigli bajt deo CRC reči ili nije. Ukoliko je pristigli bajt deo CRC reči, prelazi se u stanje sWait u kome se čita cela CRC reč. Ukoliko to nije slučaj, pristigli validni podaci nisu deo CRC reči i ostaje se u stanju sCRC. U slučaju da podaci nisu validni, prelazi se u stanje sIdle pri čemu je trenutna transakcija završena. Tabela III sadrži broj potrebnih bajtova koje je potrebno pročitati u stanju sWait kako bi se pročitala cela CRC reč dužine 32 bita.

Tabela III
BROJ BAJTOVA KOJE JE POTREBNO PROČITATI U STANJU SWAIT.

| regData[31:0] | Detektovana širina reči | Broj bajtova koje je potrebno učitati |
|---|---|---|
| 0xFFFF00FF | 16 bita | 3 |
| 0xFFFF03F0 | 12 bita | 3 |
| 0xFFF03F03 | 12 bita | 4 |
| 0xFFFF01FC | 14 bita | 3 |
| 0xFFC07F01 | 14 bita | 4 |
| 0xFFF01FC0 | 14 bita | 4 |
| 0xFFFC07F0 | 14 bita | 4 |

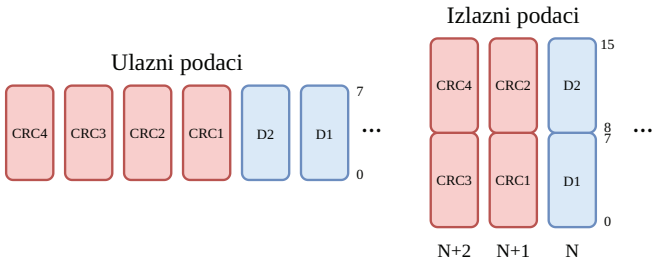
Izlaz bloka *Detekcija Reči* je signal out_word širine tri bita koji predstavlja detektovanu širinu reči. Ukoliko je bit najveće težine jednak jedinici, nije detektovana validna širina reči. Veza između vrednosti signala out_word i detektovane širine reči je data u tabeli IV.

Tabela IV
VEZA IZMEĐU SIGNALA OUT_WORD I ŠIRINE DETEKTOVANE REČI.

| Detektovana reč | 16 | 12 | 14 | CRC | greška |
|---|---|---|---|---|---|
| out_word | 3b'000 | 3b'001 | 3b'010 | 3b'011 | 3b'1xx |

### D. Pakovanje osmobitnih podataka u podatke širine 16 bita

*Bajt u Reč* blok sa slike 2 pakuje podatke širine 8 bita u reči širine 16 bita. Takođe, ukoliko radarski senzor pošalje CRC podatak širine 32 bita, on se može naći samo na kraju transakcije. Pakovanje podataka se izvršava na osnovu



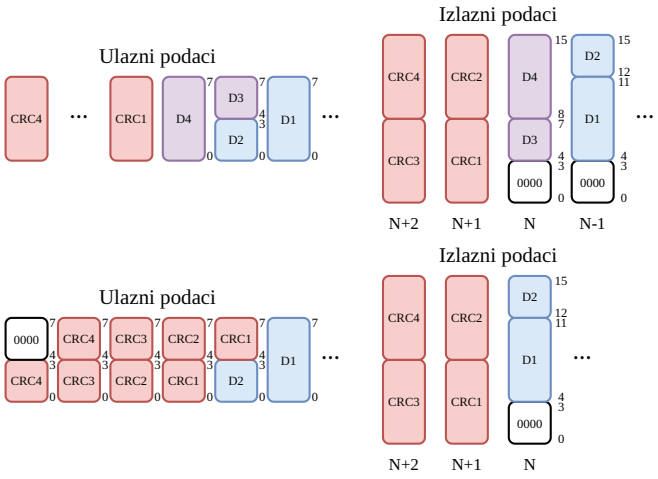

Slika 9. Primer pakovanja podataka kada radarski senzor šalje podatke širine 16 bita i CRC širine 32 bita. Pristigli podaci koji čine jednu reč obojeni su istom bojom.

širine reči detektovane u bloku *Detekcija Reči* i rednog broja pristigle reči.

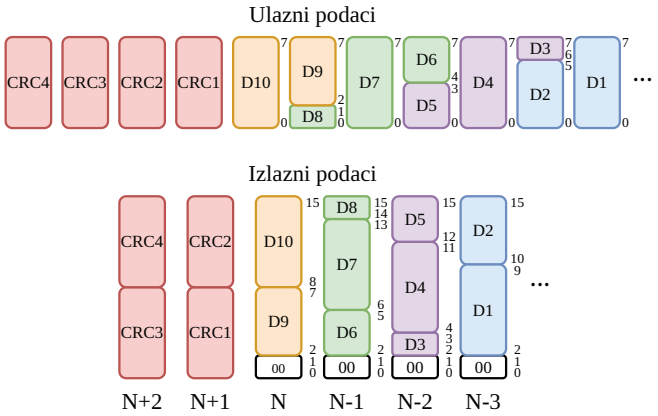
Na slici 9 prikazana je situacija kada radarski senzor šalje podatke širine 16 bita i CRC širine 32 bita. Opciona CRC reč, koja je širine 32 bita, se šalje kao dve uzastopne reči širine 16 bita.

U slučaju podataka širine 12 bita, u zavisnosti od broja poslatih reči postoje dve situacije, pri čemu je oba slučaja na kraju transakcije poslat CRC. U prvoj situaciji broj poslatih reči je deljiv sa dva i ovo je prikazano na slici 10 a). Situacija kada broj poslatih reči nije deljiv sa dva je prikazana na slici 10 b). Ovaj slučaj je komplikovaniji od prethodnog s obzirom na to da CRC podaci nisu poravnati na 8 bita.

U slučaju slanja podataka širine 14 bita, postoji četiri različitih situacija. Najjednostavniji slučaj je kada radarski senzor pošalje $4K$ reči , pri čemu je $K$ ceo broj, i na kraju transakcije CRC. U ovom slučaju, CRC je poravnat na 8 bita. U ostalim slučajevima, CRC nije poravnjan i potrebno je voditi računa o pakovanju pristiglih CRC bajtova. Na slici 11 je ilustrovana situacija kada je broj primljenih reči deljiv sa 4.




Slika 10. Primer pakovanja podataka kada radarski senzor šalje podatke širine 12 bita i CRC širine 32 bita. Pristigli podaci koji čine jednu reč obojeni su istom bojom. Pod a) je prikazan slučaj kada je broj primljenih reči deljiv sa dva dok pod b) broj primljenih reči nije deljiv sa dva.




Slika 11. Primer pakovanja podataka kada radarski senzor šalje podatke širine 14 bita i CRC širine 32 bita. Ulazni podaci su prikazani u gornjem delu slike dok su zapakovani podaci prikazani u donjem delu slike slike. Pristigli podaci koji čine jednu reč obojeni su istom bojom.

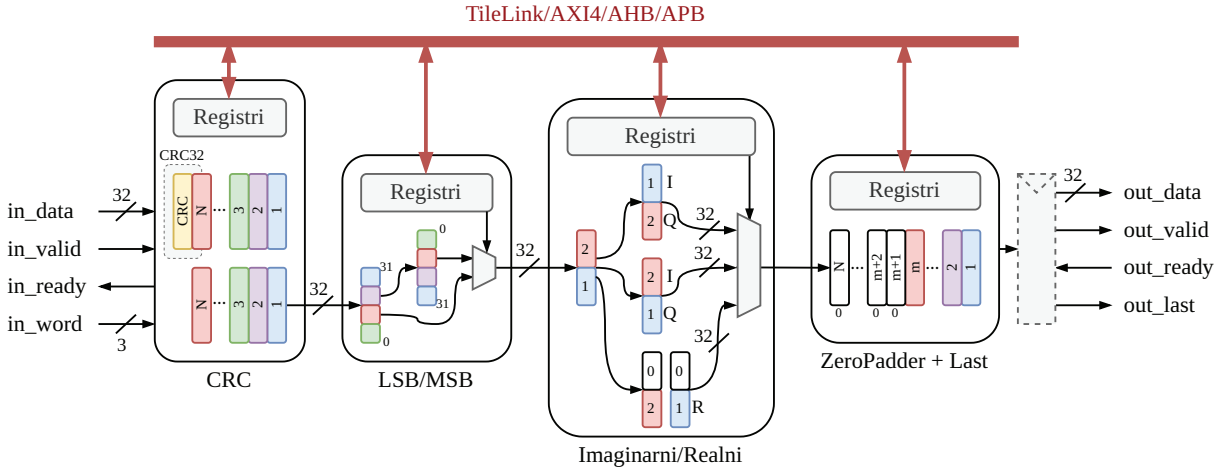
### *E. Transformacija podataka*

Radarski senzori mogu poslati realne ili kompleksne podatke. U slučaju slanja kompleksnih podataka, prvo se može poslati realni deo podatka pa imaginarni i obrnuto. Takođe, radarski senzor može poslati preko LVDS linija prvo bit najveće težine (eng. *Most Significant Bit* – MSB) ili bit najmanje težine (eng. *Least Significant Bit* – LSB). Takođe, 32-bitni CRC podatak je moguće poslati na kraju paketa podataka.

Kako bi lanac za digitalnu obradu signala primio podatke u odgovarajućem formatu, potrebno je izvršiti transformaciju pristiglih radarskih podataka. Blok *Transformacija* sa slike 2 vrši ovu funkciju i njegov uprošćeni blok dijagram je dat na slici 12.

S obzirom na to da je ulaz u blok za transformaciju podataka širine 16 bita a radarski podaci mogu biti širine 12, 14 ili 16 bita, ukoliko je reč kraća od 16 bita najniža dva ili četiri bita moraju biti popunjeni nulama kao što je to prikazano na slikama 10 i 11. Ulazni AXI4-Stream interfejs bloka za transformaciju podataka je proširen dodatnim signalom `in_word` koji nosi informaciju o detektovanoj širini reči. Izlazni interfejs sadrži `o_last` signal koji označava poslednji podatak u paketu.

Kako bi ovaj blok pravilno transformisao podatke, u njegove kontrolne registre je potrebno upisati da li su pristigli podaci realni ili kompleksni, da li se prvo šalje realni ili imaginarni deo reči, kao i to da li se šalje prvo MSB ili LSB bit. Blok za transformaciju podataka na početku upoređuje pristigli CRC podatak sa proračunatom vrednošću CRC-a. Ukoliko se vrednosti ne poklapaju, u odgovarajući statusni registar se upisuje logička jedinica koji označava da pristigli podaci nisu validni tj. da postoji greška u njima. Nakon ovoga se primljeni CRC uklanja iz paketa podataka. Zatim, u zavisnosti od toga da li je prvo poslat MSB ili LSB vrši se rotiranje bita i ekstenzija znaka. Dalje, na osnovu toga da li su primljeni podaci realni ili kompleksni (i ukoliko su kompleksni, da li je poslat prvo realni ili imaginarni deo reči) vrši se pakovanje




Slika 12. Uprošćeni blok dijagram modula za transformaciju radarskih podataka. Na izlazu je moguće generisati AXI4-Stream bafer (blok nacrtan isprekidanom linijom) proizvoljne dubine.

podataka u 32-bitne reči. Ukoliko su pristigli podaci realni, gornjih 16 bita se popunjava nulama. Na kraju, na osnovu vrednosti kontrolnog registra koji označava broj reči u jednom paketu podataka i registra koji označava širinu FFT prozora, vrši se popunjavanje nulama (eng. *zero padding*) i generisanje signala `o_last`.

### *F. FPGA implementacija*

Predloženi sistem za prijem podataka sa FMCW radarskog senzora je implementiran kao blok za akviziciju FMCW radarskih podataka u više različitih sistema za obradu signala [7], [8]. U tabeli V je prikazano zauzeće resursa na Digilent Nexys Video FPGA razvojnoj platformi u slučaju kada se podaci šalju samo preko jedne diferencijalne linije ($N = 1$) i kada se šalju preko četiri linija ($N = 4$). Učestanost LVDS taktnog signala je u obe situacije 450 MHz.

Tabela V
ZAUZEĆE RESURSA DIGILENT NEXYS VIDEO ARTIX-7 FPGA RAZVOJNE PLOČE ZA RAZLIČIT BROJ KANALA $N$ U SLUČAJU KADA JE PLL INSTANCIRAN.

| Resursi | N = 1 | N = 4 | Dostupno |
|---|---|---|---|
| LUT (Logic) | 1015 | 2757 | 134600 |
| LUT (Memory) | 1444 | 5692 | 46200 |
| LUT (DRAM) | 1420 | 5644 | |
| LUT (Shift Reg.) | 24 | 48 | |
| Slice Registers | 811 | 2163 | 269200 |
| F7 Muxes | 128 | 512 | 67300 |
| F8 Muxes | 32 | 128 | 33650 |
| PLL | 1 | 1 | 10 |

## IV. ZAKLJUČAK

U ovom radu prikazan je sistem za prijem i pretprocesiranje podataka sa FMCW radarskih senzora preko LVDS interfejsa. Projektovani sistem deserijalizuje primljene podatke, vrši automatsku sinhronizaciju podataka i detekciju širine radarske reči kao i pristiglog CRC podatka. Pored ovoga, projektovani sistem izračunava CRC na osnovu pristiglih podataka i upoređuje ga sa pristiglim CRC podatkom čime se obezbeđuje integritet pristiglih podataka. Vrši se formatiranje podataka u zavisnosti od toga da li je radarski senzor poslao realne ili komplekse podatke, kao i na osnovu toga da li je prvo poslat bit najviše ili najniže težine nakon čega su podaci spremni za dalje procesiranje.

Projektovani sistem je implementiran i kao zaseban blok a i korišćen je u okviru složenijih sistema za obradu FMCW radarskih signala čime je pokazana njegova primena u sistemima za obradu radarskih signala u realnom vremenu.

## LITERATURA


[1] D. Watzenig i M. Horn, *Automated driving: safer and more efficient future driving*. Springer, 2016.

[2] Li, Xinrong and Wang, Xiaodong and Yang, Qing and Fu, Song, "Signal processing for TDM MIMO FMCW millimeter-wave radar sensors", *IEEE Access*, sv. 9, str. 167 959–167 971, 2021.

[3] Texas Instruments, *Understanding Serial LVDS Capture in High-Speed ADCs*, 2013.

[4] Texas Instruments, *AWR2243 Single-Chip 76 to 81 GHz FMCW Transceiver*.

[5] J. Bachrach, H. Vo, B. Richards i dr., "Chisel: constructing hardware in a scala embedded language", u *Proceedings of the 49th Annual Design Automation Conference*, 2012., str. 1216–1225.

[6] AMD, *Vivado Design Suite 7 Series FPGA and Zynq 7000 SoC Libraries Guide – UG953*, 2024.

[7] N. Petrović, M. Petrović i V. Milovanović, "Radar Signal Processing Architecture for Early Detection of Automotive Obstacles", *Electronics*, sv. 12, br. 8, str. 1826, 2023.

[8] V. Milovanović, M. Petrović, V. Damnjanović i dr., "A Rocket Core with Radar Signal Processing Accelerators as Memory-Mapped I/O Devices", *RISC-V Summit Europe*, 2024.