UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

IMPLEMENTACIJA SERIJSKOG INTERFEJSA ZA KOMUNIKACIJU RAZVOJNE PLOČE I RAČUNARA

Master rad

Mentor:
Dr Zoran Čiča, docent

Kandidat:
Jelena Radulović 3332/2014

Beograd, Septembar 2016.

SADRŽAJ

SADRŽAJ **2**

1. UVOD **3**

2. RS-232 SERIJSKI PROTOKOL **4**

2.1. DEFINICIJA I OSOBINE RS-232 PROTOKOLA 4
2.1.1. Serijski prenos podatka *4*
2.1.2. Asinhroni protokoli *5*
2.1.3. Frekvencija prenosa (baud rate) *6*
2.2. FIZIČKA LINIJA VEZE 6
2.2.1. RS-232 Bit stream *6*
2.2.2. Provera parnosti *7*
2.3. PRIMENA RS-232 PROTOKOLA 8
2.3.1. RS-232 konektori za PC računare *8*

3. IMPLEMENTACIJA RS-232 PROTOKOLA **11**

3.1. RS-232 PREDAJNIK 11
3.1.1. Ulazni i izlazni parametri modula *11*
3.1.2. Konačni automat *12*
3.2. RS-232 PRIJEMNIK 16
3.2.1. Ulazni i izlazni parametri modula *16*
3.2.2. Konačni automat *16*
3.2.3. Top-level RS-232 modul *21*

4. OPIS PERFORMANSI I VERIFIKACIJA DIZAJNA **22**

4.1. OPIS PERFORMANSI 22
4.2. VERIFIKACIJA DIZAJNA 22

5. ZAKLJUČAK **27**

LITERATURA **28**

1. UVOD

Zahtevno tržište komunikacija između industrijskih uređaja, kao i veliki broj proizvođača mrežne opreme su glavni uzroci velikog broja različitih industrijskih komunikacionih mreža koje se danas koriste. Za ostvarivanje veze između uređaja različitih proizvođača potrebno je da ti uređaji imaju iste komunikacione interfejse i procedure za razmenu podataka, što je uslovilo standardizaciju velikog broja protokola, kako za paralelni, tako i za serijski prenos podataka.

U ovom radu je predstavljena hardverska implementacija jednog od serijskih komunikacionih protokola. RS-232 je prvi standardni protokol za serijski prenos podataka (*serial interface*), koji se koristi i danas, pre svega u industriji i za potrebe izrade mrežne opreme. Električne, fizičke i funkcionalne karkteristike ovog interfejsa su standardizovane od strane EIA (*Electronics Industries Aliance*). RS-232 protokol je kreiran za relativno sporu komunikaciju jednog *data* terminala (PC računar) sa nekim perifernim komunikacionim uređajem, kao što je npr. modem, koji se nalazi na relativno malom rastojanju od terminala.

RS-232 protokol je moguće implementirati na razvojnu FPGA (*Field Programmable Gate Array*) ploču, a zatim slati jednostavne komande sa PC računara ka FPGA ploči uz pomoć implementiranog protokola. Za realizaciju implementacije koristili smo Verilog programski jezik, a razvoj i testiranje dizajna su izvršeni u ISE razvojnom okruženju za čipove proizvođača Xilinx. Kod projekta će biti dati u elektronskoj formi na priloženom CD-u. Rezultat rada, pored implementacije pomenutog algoritma, je i verifikacija i analiza performansi implementacije.

Ostatak rada je organizovan na sledeći način: u drugom poglavlju je data definicija RS-232 protokola, objašnjene su njegove osnovne karakterisitike, kao i mogućnost primene. Treće poglavlje sadrži opis realizovane implementacije. Opisani su neophodni entiteti dizajna, uz prikaz relevantnih delova programskog koda. U četvrtom poglavlju dat je pregled performansi i upotrebljenih resursa na čipu. Peto poglavlje predstavlja zaključak rada i sadrži subjektivne utiske autora o realizovanom protokolu.

2. RS-232 SERIJSKI PROTOKOL

RS-232 je prvi standardni protokol za serijski prenos podataka. U ovom poglavlju će biti reči o definiciji i osobinama RS-232 protokola, kao i o karakteristikama asinhronog i serijskog prenosa. Detaljno je opisan prenos podataka preko fizičke linije veze kada se koristi RS-232 standard. U poslednjem odeljku su nabrojane neke od najznačajnijih primena standarda i prikazan je izgled serijskog konektora PC računara.

2.1. Definicija i osobine RS-232 protokola

RS-232 protokol u potpunosti definiše jednu vrstu asinhrone serijske komunikacije, odnosno tip, strukturu i moguće brzine prenosa serijske poruke. Standard takođe definiše i fizički nivo prenosa poruke, naponske nivoe na liniji u toku prenosa poruke, kao i potreban hardver. Svi RS-232 primopredajnici moraju da rade u skladu sa ovim standardom [1].

Neki od osnovnih nedostataka RS-232 protokola su: relativno malo rastojanje na koje se podaci mogu preneti (do 15m), relativno mala brzina prenosa (do 20kb/s) i mogućnost povezivanaja samo jednog predajnika i prijemnika. Sa razvojom tehnologije i mrežnih infrastruktura, razvila se i potreba za vidovima komunikacije koji obuhvataju više primopredajnika na raznim udaljenostima, te su standardizovani i novi protokoli (RS-422 i RS-485).

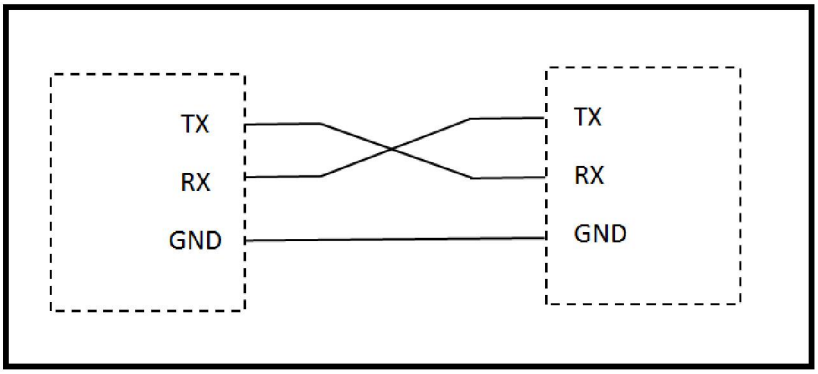
2.1.1. Serijski prenos podatka

Podaci se kroz medijum (žicu, vod, magistralu itd.) prenose ili serijski ili paralelno. Serijski prenos podataka podrazumeva prenos svih bita podataka preko iste linije prenosa, ali u različitim vremenskim trenucima odnosno u vidu vremenske sekvence, bit po bit. Odlika serijskog prenosa je jednostavnost realizacije. Kod paralelnog prenosa, n bita (n = 8, 16, 32, 64) podataka prenosi se istovremeno. Paralelni prenos podataka je brži od serijskog, ali je skuplji (zahteva veći broj linija).

Serijski komunikacioni protokoli predstavlja precizno definisane procedure i sekvence bita, karaktera i upravljačkih kodova korišćene za prenos preko komunikacione linije. S obzirom da je danas tržište veoma zahtevno i da postoji veliki broj proizvođača, imamo veliki broj različitih industrijskih komunikacionih mreža i protokola koji se koriste. Kao što je već spomenuto, RS-232 je prvi protokol koji je standardizovan za serijski prenos.

Prema RS-232 standardu, informacija se šalje u vidu niza bita preko fizičke linije veze. Biti informacije su grupisani u vidu digitalnih reči i RS-232 protokol dozvoljava promenljivu dužinu reči, od 5 i 8 bita. Sa toliko bita se prenosi informacija u jednoj poruci ili paketu. Veoma je važno da se i prijemnik i predajnik podese na isti broj bita, inače dolazi do grešaka prilikom rada protokola. Standard još definiše i dodatne bite za sinhronizaciju i detekciju greške, kako bi se ostvario bolji prenos, međutim prisustvom sinhronizacionih bita ujedno se gubi na vremenu i smanjuje se propusni opseg.

S obzirom na to da postoje posebne linije za prijem i predaju podataka, dva primopredajnika mogu istovremeno da šalju i primaju podatke. Linije su potpuno nezavisne, tako da nije važno ko je prvi započeo prenos. Na ovaj način dobijamo *full duplex* vezu, u kojoj su oba primopredajnika od jednakog značaja. S druge strane, neki primopredajnici ne mogu istovremeno da šalju i primaju podatke. U tom slučaju se projektuje *half duplex* veza, što znači da je komunikacija dvosmerna ali ne potpuno, jer se odgovor šalje tek nakon prijema upita. U ovom sistemu uglavnom postoji jedan master uređaj na liniji i on započinje komunikaciju, a zatim čeka odgovor od slejv uređaja. Potpuna asinhrona serijska veza dva uređaja (*full duplex*) koji mogu istovremeno da primaju i šalju podatke, može se realizovati sa samo tri linije, što je prikazano na slici 2.1.1.



Slika 2.1.1. Potpuna bidirekcionalna veza

2.1.2. Asinhroni protokoli

RS-232 protokol pripada grupi asinhronih protokola. Protokol se smatra asinhronim ukoliko predajnik ne prosleđuje signal takta (*clock* signal) prijemniku, već samo podatke koji se menjaju u vremenu. Kod asinhronih protokola sinhronizacija postoji samo u periodu kada se obavlja prenos. Takođe, postoji samo linija za prenos podataka, ne i takt linija, pa zato mora postojati neki drugi vid sinhronizacije između prijemnika i predajnika.

Standardom se definiše način detekcije i kraja poruke, kao i sinhronizovano čitanje poruke od strane prijemnika. Predajnik može u bilo kojem trenutku da generiše bite, pa prijemnik mora da poseduje mehanizam na osnovu koga će znati u kom trenutku da prima podatke. Sa ciljem da se odredi korektni početak rada prijema tipično se koristi metod rada poznat kao start-stop, o kome će biti više reči u narednim odeljcima [2].

Do greške najčešće dolazi ako pogrešno sinhronizovani prijemnik počne u pogrešno vreme da čita poruku. U tom slučaju sinhronizacija se mora ponovo uspostaviti, čime se troši dodatno vreme. Asinhroni prenos se i dalje najčešće koristi jer je jeftiniji (ne realizuje se *clock* signal) i otporniji je na šum, ali za veoma brze prenose je sinhrona komunikacija bolje rešenje.

Sinhronizacija se ostvaruje pomoću frekvencije prenosa (*baud rate*) koja predstavlja broj bita poslatih u sekundi. *Baud rate* je takođe definisan od strane standarda, i mora biti unapred isto podešen i za prijemnik i za predajnik, inače dolazi do greške u prenosu.

2.1.3. Frekvencija prenosa (baud rate)

Baud rate je mera kojom se označava brzina prenosa kod asinhrone komunikacije i pokazuje koliko je bita moguće poslati preko serijskog linka u jedinici vremena. Ako je *baud rate* 1200 to znači da možemo poslati 1200 bita u sekundi, odnosno da je trajanje prenosa jednog bita jedna milisekunda.

Prilikom implementacije RS-232 interfejsa nije dozvoljeno koristiti bilo koje vrednosti za frekvenciju prenosa, već samo one definisane standardom. Neke od tipičnih definisanih vrednosti su: 1200, 9600, 38400, 115200 bauda.

Primera radi, ako je *baud rate* 115200 bauda, to znači svaki bit traje (1/115200) = 8.7μs, dok je za prenos osmobitnih podataka potrebno 8 x 8.7μs = 69μs. Međutim, kako je za svaki bajt podataka neophodno poslati start bit, kao i stop bit, za prenos jednog bajta podataka je zapravo potrebno 10 x 8.7μs = 87μs, što dalje rezultira maksimalnom brzinom od 11.5kB/s. Takođe, za frekvenciju prenosa od 115200 bauda, neki spori uređaji zahtevaju duže trajanje stop bita (dužina od 1.5 ili 2 bita), što smanjuje maksimalnu moguću brzinu na oko 10.5kB/ s.

Prilikom realizacije RS-232 protokola neophodno je implementirati i tzv. *baud rate* generator odnosno izvršiti deljenje frekvencije. Ukoliko je frekvencija takt signala sistema 50MHz, a *baud rate* 115200 bauda, onda je za prenos jednog bita neophodan vremenski period od 50MHz/115200 = 434 perioda *clock* signala. Kada je poznata *baud rate* vrednost poznati su i vremenski intervali u kojima je moguće očekivati prijem podataka ili vršiti slanje – za prenos svakog bita se čeka 434 perioda *clock* signala.

2.2. Fizička linija veze

RS-232 protokol dozvoljava samo dva fizička stanja na liniji za prenos. Prvi naponski nivo je -12V na liniji, koji predstavlja ON stanje ili broj 1 ili marker. Ovo stanje predstavlja neaktivno stanje asinhrone linije. Drugi naponski nivo je +12V na liniji i predstavlja OFF stanje ili broj 0 ili prazno mesto.

Mikroprocesori koji šalju ili primaju signale rade sa napajanjem od 5V i sposobni su da generišu naponske nivoe od 0V (predstavlja logičku 0 ili broj 0) i 5V (logička 1 ili broj 1). RS-232 prijemni i predajni drajveri menjaju nivo i invertuju logiku ovih signala tako da logička 1 na RS-232 liniji postaje -12V, dok logička 0 na RS232 liniji postaje +12V.

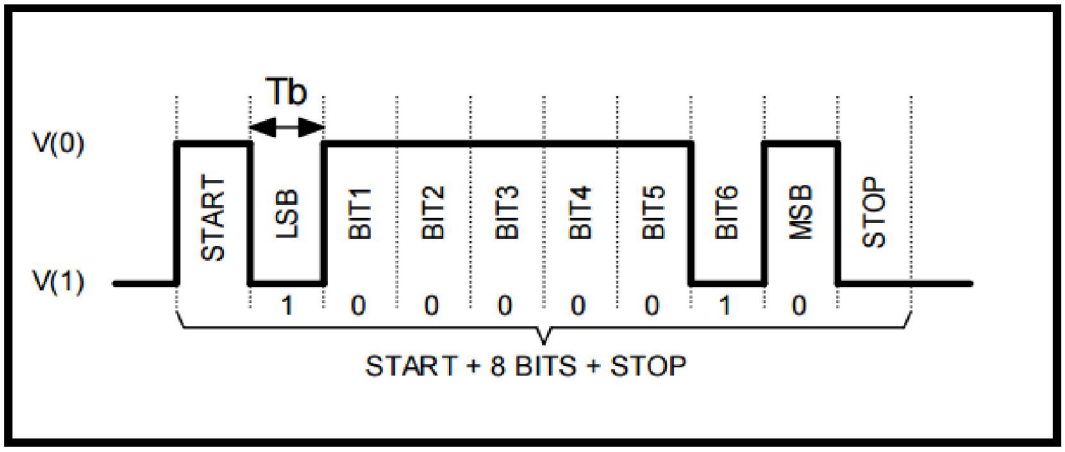
Kada se ne prenose podaci linija je neaktivna i na naponskom nivou od -12V, što predstavlja stanje logičke 1 ili binarni broj 1. Za razliku od neaktivnih -12V, prvi bit u poruci je start bit i uvek je naponskog nivoa +12V. Na ovaj način se signalizira početak poruke.

2.2.1. RS-232 Bit stream

Slika 2.2.2 prikazuje prenos bita preko linije RS-232 veze (*bit stream*). Kao što se može uočiti, prenos podataka se uvek započinje slanjem start bita. Start bit je uvek logički nivo 0, da bi se razlikovao od logičke 1 tj. neaktivnog stanja. Prijemnik čitanjem start bita detektuje početak prenosa. Međutim, kada pročita start bit neophodno je da odredi trenutak u kome čita naredni bit. Za ovaj proračun se koristi frekvencija prenosa (*baud rate*). Kao što je već objašnjeno, *baud rate* predstavlja broj bita poslatih u jednoj sekundi. Kod asinhronog prenosa on se unapred definiše i vrlo je bitno da i prijemnik i predajnik koriste isti *baud rate* kako bi se izbegla greška prilikom prenosa.

Nakon slanja start bita, predajnik započinje prenos bita podataka (*data* bita). Bit najmanjeg značaja (LSB – *least significant bit*) se uvek prvi šalje. Ukoliko se šalje bit 1 naponski nivo na RS232 liniji je -12V, dok slanje bita 0 dovodi do naponskog nivoa od +12V. [1]

Kada je završen prenos *data* bita, okvir poruke se zatvara slanjem stop bita koji ima vrednost logičke 1. U slučaju da je prijemnik pogrešio u sinhronizaciji, stop bit daje novu šansu za resinhronizaciju. Ako prijemnik detektuje logičku 0 na mestu stop bita, on uočava grešku i prekida prijem poruke. Ovakva greška se naziva *framing error* – greška okvira i pokazuje da biti informacije nisu unutar definisanog okvira, okruženi stop i start bitima. Nakon greške okvira, ponovo se vrši resinhronizacija prijemnika.



Slika 2.2.2 RS-232 prenos bita

Prijemnik prati okvir poruke i očekuje start i stop bite u tačnim vremenskim razmacima koje definiše *baud rate*. Na taj način prijemnik prepoznaje i *baud rate* pristigle poruke i nakon određenog vremena se sinhronizuje na tu novu brzinu transfera. Procedure za *baud rate* sinhronizaciju uvek izbegavaju slanje svih 0 u informacionom delu paketa, jer prijemnik može da ih pomeša sa stop bitom. Prema RS-232 standardu, stop bit može imati više bitskih dužina. Ako je stop bit duži od ostalih bita, on ujedno definiše i minimalno vreme u toku koga linija mora da bude u *idle* stanju tj. neaktivna. Ovaj deo standarda predstavlja podršku za spore uređaje. Trajanje stop bita se tipično postavlja na dužinu od 1, 1.5 ili 2 bita.

2.2.2. Provera parnosti

Bit parnosti (*parity* bit) služi za eventualnu detekciju greške i može se ugraditi u RS-232 poruku, nakon slanja zadnjeg bita podataka odnosno MSB (*most significant bit*) bita, a pre stop bita. Ovaj bit postavi predajnik, tako što unapred proveri broj jedinica u poruci koju šalje. RS-232 veza može da radi u *even parity* modu ili u *odd parity* modu. Za slučaj *even parity* tj. parnog bita parnosti, bit parnosti se postavlja na 1 ako je broj jedinica u informacionom delu poruke (bez start i stop bita) paran, i na 0 ako je broj jedinica neparan. Za slučaj *odd parity* odnosno neparnog bita parnosti, logika je obrnuta.

Za pravilan rad interfejsa, neophodno je da i prijemnik i predajnik imaju podešen isti tip parnosti. Ovo nije savršen način za detekciju greške s obzirom da u slučaju parnog broja grešaka na prijemu prijemnik dobija isti bit parnosti. Provera bita parnosti je najčešće nedovoljna provera

tačnosti prenosa, pogotovo u sredinama sa izraženim šumom, stoga se koriste razne CRC i LCRC metode za proveru tačnosti celokupne RS-232 poruke, nakon što su svi RS-232 paketi primljeni. Osim ovih metoda, moguće je primeniti protokole višeg nivoa koji u sebi imaju ugrađene mehanizme za detekciju greške u poruci.

2.3. Primena RS-232 protokola

Iako je danas razvijen velik broj novih standarda, RS-232 se i danas često koristi, pogotovo za prenos na relativno kratkim rastojanjima. Neke od najčešćih primena su:

- Programiranje PLC i mikroprocesora preko PC računara
- Parametrizacija uređaja
- Rad sa modemom, štampačem i sličnim perifernim uređajima

Osnovni razlog za primenu RS-232 standarda jeste njegova jednostavost. On omogućava korisniku direktnu komunikaciju kroz serijske portove. U oblastima kao što su automatika, industrija, laboratorijska istraživanja postoji velika potreba za primenom RS-232 protokola zbog rada sa skupom, ali zastarelom opremom. Daleko je jeftinije koristiti RS-232 protokol za povezivanje opreme sa nekim novim komponentama, nego je potpuno zameniti novom. Takođe, neki *embedded* sistemi koriste RS-232 serijske portove za komunikaciju.

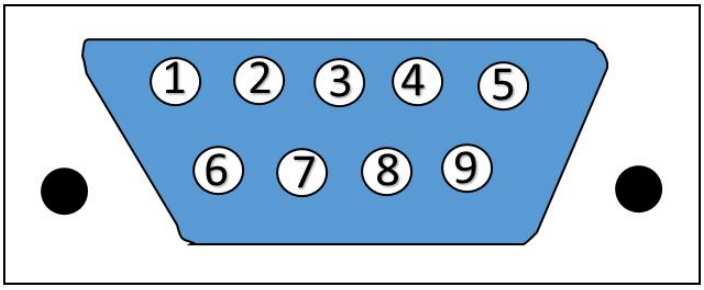
S druge strane, kada se javila potreba za vidovima komunikacije koji obuhvataju više primopredajnika na različitim udaljenostima, pokazali su se i nedostaci RS-232 protokola. Podaci se mogu razmenjivati na malim rastojanjima, tipično do 15m, brzina prenosa je relativno mala (do 20kb/s) i moguće je povezati samo jedan predajnik i prijemnik. Za prenos podataka na većim rastojanjima i za veće brzine standardizovan je RS-422 protokol.

2.3.1. RS-232 konektori za PC računare

Serijski port, koji se takođe naziva i komunikacijski (*communication* - COM) port, je bidirekcioni port. Bidirekciona komunikacija omogućuje uređaju i da prima i da šalje podatke.

RS-232 konektor za PC računare je originalno kreiran sa 25 pinova (DB25). Razlog za to je bio mogućnost povezivanja dva serijska kanala na isti konektor. U praksi se pokazalo da se uobičajeno povezuje samo jedan serijski kanal tako da je 9-pinski konektor za serijski vezu postao standard.

DB9 serijski konektor je prikazan na slici 2.3.1 Pin br.2 predstavlja prijemni pin (*Receive data - Rx*), pin br.3 je predajni pin (*Transmit data - Tx*), a signalno uzemljenje se nalazi na pinu br.5. Uzemljenje je povezano na kućište konektora i spoljni omotač. Ovi pinovi su dovoljni za prostu asinhronu komunikaciju dva uređaja koja se oslanja samo na programsku sinhronizaciju. Ostali pinovi DB9 konektora služe za hardverski *handshake* za rad sa modemom [3].



Slika 2.3.1 Izgled DB9 serijskog konektora

Handshake je proces tokom koga PC računar šalje signal modemu ili nekom drugom perifernom uređaju u cilju uspostavljanja veze pre samog prenosa podataka. Pre nego što započne prenos podataka preko fizičke linije veze, predajna i prijemna strana podešavaju komunikacione parametre. *Handshake* mehanizam omogućava samostalno povezivanje relativno heterogenih sistema ili opreme, tako da nije neophodno da čovek ručno podesi parametre komunikacije. Nazivi svih pinova su datu u tabeli 2.3.1.1.

Tabela 2.3.1 Pinovi serijskog DB9 konektora

BROJ PINA	IME SIGNALA	ZNAČENJE SIGNALA
1	*Data carrier detect* (CD)	Periferni uređaj signalizira terminalu da je veza uspostavljena
2	***Receive data* (Rx)**	Serijski ulaz podataka
3	***Transmit data* (Tx)**	Serijski izlaz podataka
4	*Data terminal ready* (DTR)	Terminal signalzira perifernom uređaju da je spreman za rad.
5	***Signal ground***	Uzemljenje sistema (masa)
6	*Data set ready* (DSR)	Periferni uređaj signalizira da je spreman za rad
7	*Request to send* (RTS)	Terminal signalizira da želi da uspostavi vezu sa perifernim uređajem
8	*Clear to send* (CTS)	Odgovor perifernog uređaja na RTS signal, nakon uspostavljene sinhronizacije
9	*Ring indicator* (RI)	Signalizira detekciju signala „zvona“ na telefonskoj liniji.

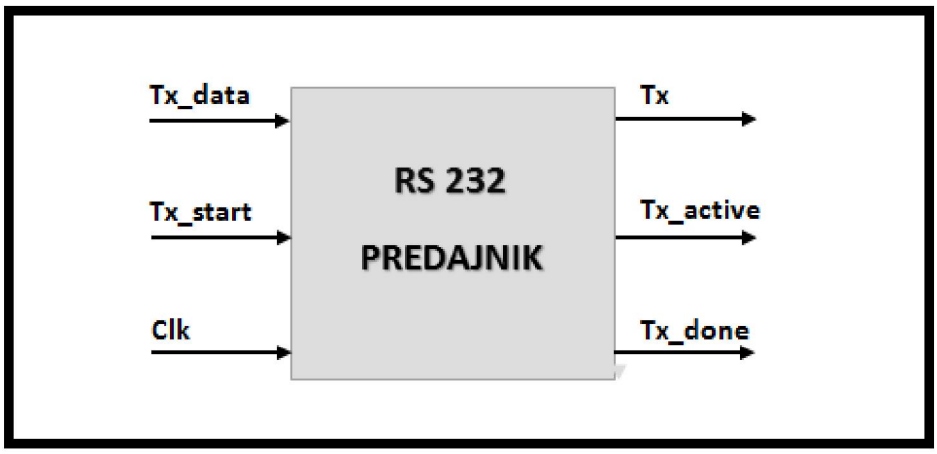
3. IMPLEMENTACIJA RS-232 PROTOKOLA

U ovom poglavlju biće objašnjen programski kod napisan za potrebe implementacije RS-232 protokola. Kod je pisan u Verilog programskom jeziku. Dizajn RS-232 protokola se sastoji od dva nezavisna Verilog modula. Prvi modul implementira RS-232 predajnik, dok drugi modul implementira RS-232 prijemnik. Oba modula su instancirana u *top level* modul dizajna i koriste zajednički takt signal koji je zapravo master takt signal FPGA ploče. *Baud rate* ima vrednost 115200 bauda. Poruku čine osmobitni podaci, start bit i stop bit. Provera parnosti se ne vrši, u sklopu paketa se ne prenosi *parity* bit.

3.1. RS-232 predajnik

3.1.1. Ulazni i izlazni parametri modula

RS-232 predajnik obavlja prenos podataka tako što vrši konverziju podataka iz paralelnog u serijski format, a zatim ih prosledi na fizičku liniju veze. Na slici 3.1.1 je prikazana šema RS-232 predajnika sa označenim ulaznim i izlaznim portovima.



Slika 3.1.1 Šema RS-232 predajnika

Ulazni portovi RS-232 predajnika su :

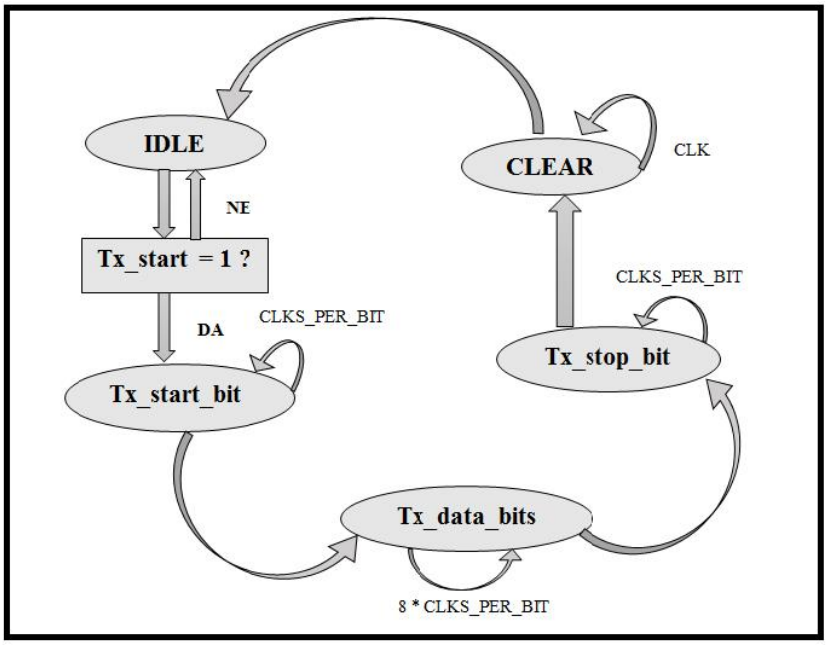
- *clk :* Označava pin na koji se vezuje takt signal FPGA ploče (frekvencija signala je f = 50MHz).
- *Tx_start* : Signal koji označava početak prenosa podataka.
- *Tx_data* : Signal koji predstavlja osmobitne podatke za prenos.

Izlazni portovi RS-232 predajnika su :

- *Tx* : Označava pin koji se povezuje na fizičku liniju za serijski prenos podataka.
- *Tx_active* : Signal čija vrednost 1 označava da je predajnik u aktivnom stanju.
- *Tx_done* : Signal koji označava da je završen prenos podataka.

3.1.2. Konačni automat

Za realizaciju prenosa podataka je najpogodnije koristiti konačan automat. Na osnovu trenutne vrednosti promenljive koja predstavlja stanje konačnog automata, primenjuje se odgovarajući kod, tj. odgovarajuća faza u prenosu poruke. Stanja koja su definisana su: *Idle, Tx_start_bit, Tx_data_bits, Tx_stop_bit* i *Clear*. Prilikom uzlazne ivice *clk* signala proverava se vrednost promenjive *State* i na osnovu njene vrednosti se vrše određene dodele. Navedena stanja su prikazana na slici 3.1.2



Slika 3.1.2 Dijagram stanja RS-232 predajnika

Stanje *Idle* je stanje u kome se dizajn odnosno RS-232 predajnik nalazi kada nema prenosa poruke. Za vreme ovog stanja, *Tx* linija se nalazi u neaktivnom stanju tj. drajvuje se bit 1. Signali *Tx_done, Tx_active* i *Tx_start* imaju vrednost 0. Sve dok je signal *Tx_start* deasertovan, predajnik se nalazi u neaktivnom stanju.

U nastavku je prikazan kod koji realizuje *Idle* stanje predajnika:

```
always @(posedge clk)
  begin
   case (State)
```

```
Idle :
 begin
  Tx  <= 1'b1;  //Tx linija se nalazi u neaktivnom stanju
  Tx_done_reg  <= 1'b0;
   Tx_active_reg <= 1'b0;  //Predajnik nije aktivan
  Clk_cnt      <= 0;
  Bit_index    <= 0;

   if (Tx_start == 1'b1) //Da li je Tx_start asertovan?
   begin
    Tx_active_reg <= 1'b1;   //Predajnik postaje aktivan
    Tx_data_reg  <= Tx_data;
    State  <= Tx_start_bit; // Prelazak u naredno stanje
   end
   else
    State <= Idle;  //Ako signal Tx_start nije asertovan
 end
```

Za početak prenosa, neophodno je asertovati signal *Tx_start*. Kada je vrednost signala *Tx_start* = 1, predajnik prelazi u sledeće stanje - *Tx_start_bit. Tx_active* signal se takođe postavlja na logičku vrednost 1 i podatke za prenos *Tx_data* smeštamo u pomoćni registar *Tx_data_reg* pre samog početka prenosa.

Tokom *Tx_start_bit* stanja se prenosi start bit, a u sledeće stanje *Tx_data_bits* se prelazi kada se prenos start bita završi. Ovaj trenutak se određuje na osnovu parametra *CLKS_PER_BIT* kojim se označava koliko je potrebno perioda takt signala da prođe, da bi se preneo jedan bit. Parametar se računa kao količnik frekvencije takt signala i *baud rate* frekvencije. U ovom konkretnom slučaju *CLKS_PER_BIT = 50MHz / 115200* , što je približno 434 *clock* perioda. Pomoćni brojač *Clk_cnt* broji do vrednosti *CLKS_PER_BIT* odnosno do završetka prenosa start bita, a zatim se resetuje i predajnik prelazi u sledeće stanje.

Kod kojim je realizovano *Tx_start_bit* stanje je dat u nastavku:

```
Tx_start_bit :
    begin
     Tx <= 1'b0;  // Šalje se start bit preko Tx linije
     //Čekati kraj prenosa start
     if (Clk_cnt < CLKS_PER_BIT-1)
     begin
       Clk_cnt <= Clk_cnt + 1;  //Inkrementiranje brojača
```

```
        State  <= Tx_start_bit;
      end
      else
      begin
      //Resetovanje brojača nakon prenosa start bita
         Clk_cnt <= 0;
         State  <= Tx_data_bits; // Prelazak u naredno stanje
      end
    end // case: Tx_start_bit
```

Stanje *Tx_data_bits* definiše period tokom kog se prenose 8-bitni podaci serijski preko linije veze *Tx*. Završetak prenosa svakog bita se ponovo određuje na osnovu brojača i *CLKS_PER_BIT* parametra. Takođe, koristi se i pomoćna promenljiva *Bit_index* u kojoj se čuva redni broj bita koji se trenutno prenosi. *Bit_index* može imati vrednosti od 0 do 7, a njegova vrednost se inkrementira, sve dok se ne prenese poslednji bit podataka.

Tx_data_bits stanje je definisano na sledeći način :

```
Tx_data_bits :
    begin
     Tx <= Tx_data_reg[Bit_index];
     //Čeka se završetak prenosa jednog bita podataka.
     if (Clk_cnt < CLKS_PER_BIT-1)
       begin
       Clk_cnt <= Clk_cnt + 1;  //Inkrementiranje brojača
       State  <= Tx_data_bits;
       end
     else
      begin
      //Reset brojača, jedan bit podataka je prenet
      Clk_cnt <= 0
      //Provera da li su poslati svi biti
      if (Bit_index < 7)
        begin
         Bit_index <= Bit_index + 1;
         State    <= Tx_data_bits;
        end
       else
```

```
        begin
         Bit_index <= 0;
         State     <= Tx_stop_bit;
        end
      end
    end       //case: Tx_data_bits
```

Za vrednost promenljive *Bit_index* = 7, predajnik prelazi u stanje *Tx_stop_bit*. Prenosi se stop bit preko fizičke linije veze, *Tx* = 1. Nakon toga se resetuju brojači, *Tx_active* signal koji je bio asertovan za vreme prenosa se deasertuje. Nasuprot njemu, *Tx_done* signal se sad asertuje i na taj način se signalizira *top* modulu da je prenos podataka završen.

Nakon *Tx_stop_bit* stanja predajnik ulazi u još jedno, poslednje stanje koje je označeno kao *Clear*. Tokom *Clear* stanja *Tx_done* signal je asertovan tokom trajanja jednog *clk* perioda. Nakon toga predajnik prelazi u neaktivno stanje.

Tx_stop_bit i *Clear* stanje su implementirani na sledeći način:

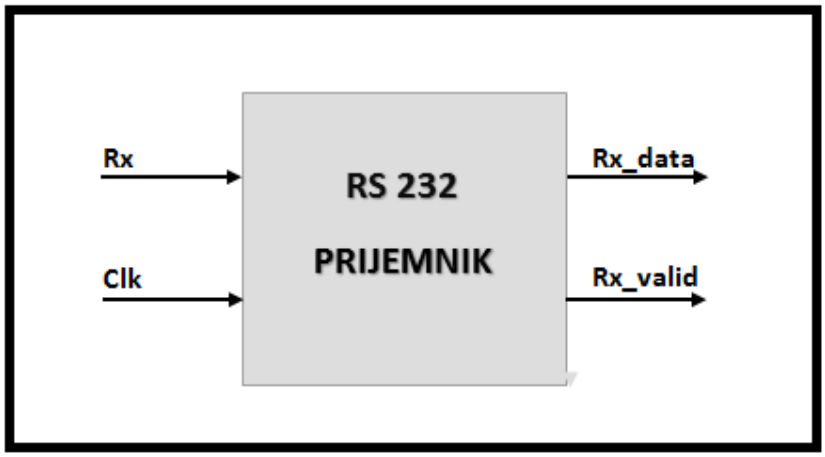
```
Tx_stop_bit :
    begin
     Tx <= 1'b1;   //Slanje Stop bita. Stop bit = 1
     //Čekanje da se završi prenos stop bita
     if (Clk_cnt< CLKS_PER_BIT-1)
       begin
       Clk_cnt <= Clk_cnt+ 1;
       State  <= Tx_stop_bit;
       end
     else
       begin
      //Signaliziranje top level modulu da je  prenos završen.
       Tx_done_reg  <= 1'b1;
       Clk_cnt    <= 0;
       State     <= Clear;
      //Prenos je završen, predajnik više nije aktivan.
       Tx_active_reg <= 1'b0;
       end
    end     //case: Tx_stop_bit

   Clear :
```

```
begin
//Na kraju se drajvuje Tx_Done signal
//u trajanju jednog clk perioda
 Tx_done_reg <= 1'b1;
 State    <= Idle;  //Povratak u Idle stanje
end
```

3.2. RS-232 prijemnik

3.2.1. Ulazni i izlazni parametri modula

RS-232 prijemnik obavlja funkciju konvertora podataka iz serijskog u paralelni format. Na slici 3.2.1 je prikazana šema RS-232 prijemnika, sa označenim ulaznim i izlaznim portovima.



Slika 3.2.1 Šema RS-232 prijemnika

Ulazni portovi RS-232 prijemnika su :

•*clk :* Označava pin na koji se vezuje takt signal FPGA ploče (frekvencija signala je *f* = 50MHz).

•*Rx* : Označava pin na koji se vezuje linija za prijem podataka.

Izlazni portovi RS-232 prijemnika su :

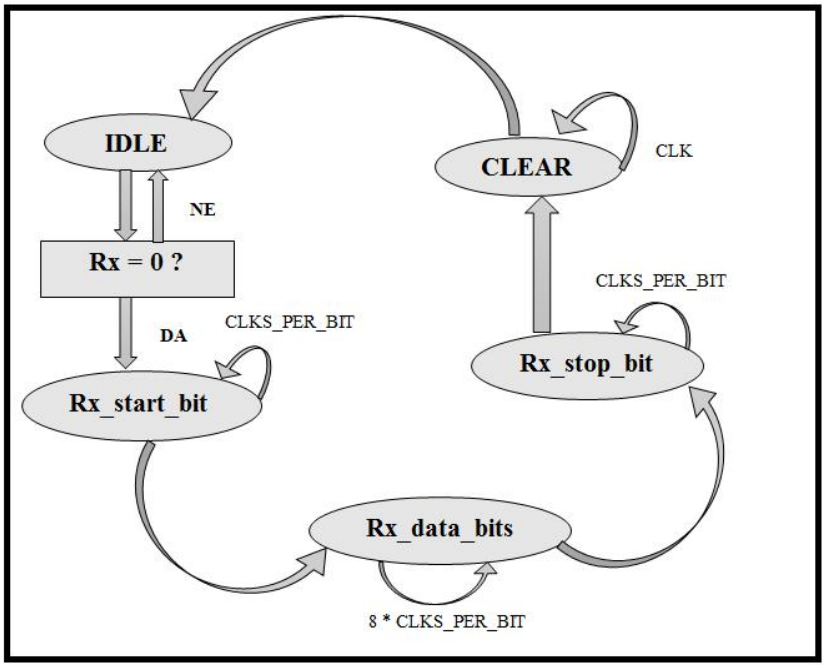
•*Rx_data* : Deserijalizovani primljeni podaci.

•*Rx_valid* : Signal koji označava da su podaci uspešno primljeni.

3.2.2. Konačni automat

Prijem podataka se realizuje na sličan način kao i prenos podataka, uz korišćenje konačnog automata i definisanje stanja prijema. Na osnovu trenutne vrednosti promenljive koja predstavlja

stanje konačnog automata, primenjuje se odgovarajući kod, tj. odgovarajuća faza u prijemu poruke. Za kreiranje konačnih automata korišćena je *case* struktura programskog jezika Verilog. Stanja koja su definisana su: *Idle*, *Rx_start_bit*, *Rx_data_bits*, *Rx_stop_bit* i *Clear*. Prilikom uzlazne ivice *clk* signala provera se vrednost promenjive *State* i na osnovu njene vrednosti se vrše određene dodele. Navedena stanja su prikazana na slici 3.2.2.1 .



Slika 3.2.2 Dijagram stanja RS-232 prijemnika

Stanje Idle je stanje u kome se RS-232 prijemnik nalazi kada nema prenosa poruke tj. sve dok je Rx linija u neaktivnom stanju. Kada prijemnik detektuje vrednost 0 tj. start bit, zna da je započeo prenos i prelazi u stanje *Tx_start_bit*.

Idle stanje je u realizovano kao:

```
always @(posedge clk)
  begin
    case (State)
     Idle :                    // Idle stanje, nema prenosa
       begin
       Rx_done  <= 1'b0;
       Clk_cnt  <= 0;
       Bit_index <= 0;
       if (Rx == 1'b0)           // Detektovan je start bit?
```

```
          State <= Rx_start_bit; // Prelazimo u naredno stanje
         else
          State <= Idle;
        end      //case: Idle
```

Kao i kod predajnika, trajanje prenosa jednog bita se određuje na osnovu parametra *CLKS_PER_BIT* i pomoćnog brojača *Clk_cnt*. Prijemnik treba da odredi kada je najbolji trenutak za odabiranje ili semplovanje (*sampling*) signala. Odabiranje signala predstavlja proces očitavanja signala u određenim vremenskim trenucima. Kako bi se dobili najbolji rezultati, odabiranje signala se vrši na sredini signala, koja se jednostavno određuje kao *CLKS_PER_BIT* / 2.

Tokom *Rx_start_bit* stanja se odvija prijem start bita. Na početku ovog stanja se vrši provera da li je *Rx* linija i dalje na naponskom nivou koji odgovara logičkoj 0. Ukoliko nije, znači da je u prethodnom koraku umesto start bita detektovan neki glič na liniji veze. Tada se prijemnik vraća u neaktivno *Idle* stanje.

Ukoliko se dizajn nalazi u *Rx_start_bit* stanju izvršava se sledeći kod :

```
Rx_start_bit :
  begin
   // Provera na sredini start bita
   if (Clk_cnt == (CLKS_PER_BIT-1)/2)
     begin
     // Da li je start bit i dalje na niskom naponskom nivou?
      if (Rx == 1'b0)
       begin
       // Reset se brojača - pronađena sredina start bita
       Clk_cnt <= 0;
       State   <= Rx_data_bits;   // Prelazak u sledeće stanje
        end
      else
       // Nije pronađen start bit, povratak u neaktivno stanje
       State <= Idle;
      end
    else
     begin
      // Brojač broji do sredine start bita
      Clk_cnt <= Clk_cnt + 1;
      State   <= Rx_start_bit;
     end
```

```
    end    // case: Rx_start_bit
```

Prijem bita podataka odvija u narednom stanju *Rx_data_bits*. Trajanje prenosa jednog bita se određuje na osnovu parametra *CLKS_PER_BIT* i pomoćnog brojača *Clk_cnt,* a odabiranje svakog bita se vrši na sredini signala. Kod je dat u nastavku:

```
Rx_data_bits :
    begin
    // Čekanje CLKS_PER_BIT-1 clk perioda za odabiranje

    if (Clk_cnt < CLKS_PER_BIT-1)
      begin
      Clk_cnt <= Clk_cnt + 1; //Inkrementiranje brojača
      State   <= Rx_data_bits;
      end
    else
      begin
      Clk_cnt         <= 0;
      Rx_byte[Bit_index] <= Rx_data_reg;
      // Provera da li je primljeno svih 8 bita podataka
      if (Bit_index < 7)
      begin
       Bit_index <= Bit_index + 1;
       State    <= Rx_data_bits;
      end
      else
      // Svi biti podataka su primljeni
      begin
       Bit_index <= 0;
       State    <= Rx_stop_bit;
      end
     end
    end    // case: Rx_data
```

Kada je vrednost promenljive *Bit_index* = 7, završen je prenos svih bita podataka i predajnik prelazi u stanje *Rx_stop_bit*. Tokom ovog stanja se odvija prijem stop bita. Nakon toga se resetuju svi pomoćni brojači, *Rx_valid* signal se sad asertuje i na taj način se signalizira *top* modulu da je prenos podataka završen.

Poslednje stanje je označeno kao *Clear*. Tokom *Clear* stanja *Rx_valid* signal je asertovan tokom trajanja jednog *clk* perioda, a zatim se vraća na naponski nivo logičke 0. Nakon toga prijemnik prelazi u neaktivno stanje. Prijem podataka je završen. Sledi kod kojim su realizovana *Rx_stop_bit* i *Clear* stanje:

```
Rx_stop_bit :
  begin
// Čekati CLKS_PER_BIT-1 clk perioda
  if (Clk_cnt < CLKS_PER_BIT-1)
  begin
    Clk_cnt <= Clk_cnt + 1;
    State   <= Rx_stop_bit;
    end
  else
    begin
     Rx_done <= 1'b1;
     Clk_cnt  <= 0;
     State    <= Clear;
    end
   end       // case: Rx_stop_bit


Clear :
  begin
   if (Clk_cnt < (CLKS_PER_BIT-1)/2)
    begin
    Clk_cnt <= Clk_cnt + 1;
    State   <= Clear;
    end
   else
    begin
    State    <= Idle;
    Clk_cnt  <= 0;
    Rx_done <= 1'b0;
    end
   end
```

3.2.3. Top-level RS-232 modul

Osim predajnika i prijemnika, kreiran je *top-level* modul koji je na najvišem hijerarhijskom nivou dizajna. U *top* modulu su kao komponente instancirani i predajnik i prijemnik. Kod kojim je ovo realizovano je dat u nastavku:

```
module RS232_top_module #(parameter CLKS_PER_BIT = 434)
  ( input clk,
   input top_Rx_line,
    input Tx_start,
    input [7:0] Tx_data,
    output Rx_done,
    output [7:0] Rx_data,
   output top_Tx_line,
    output Tx_done,
    output Tx_active
     );

  //Instanciranje RS232 prijemnika i predajnika
  RS232_Rx #(.CLKS_PER_BIT(CLKS_PER_BIT)) Rx_Inst
    (.clk(clk),
    .Rx(top_Rx_line),
    .Rx_valid(Rx_done),
    .Rx_data(Rx_data)
    );

   RS232_Tx #(.CLKS_PER_BIT(CLKS_PER_BIT)) Tx_Inst
    (.clk(clk),
    .Tx_start(Tx_start),
    .Tx_data(Tx_data),
    .Tx_active(Tx_active),
    .Tx(top_Tx_line),
    .Tx_done(Tx_done)
    );
endmodule
```

4. OPIS PERFORMANSI I VERIFIKACIJA DIZAJNA

4.1. Opis performansi

Izvršavanjem procesa analize i sinteze top level dizajna dobijene su vrednosti prikazane u tabeli 4.1. Proces analize i sinteze izvršen je u ISE razvojnom okruženju za FPGA čipove kompanije Xilinx. Procene vrednosti su date u tabeli 4.1.1. Za implementaciju je izabran uređaj XC3S500E-4VQ100 familije Spartan 3E.

Tabela 4.1.1 Procenjene performanse

NAZIV	UTROŠENI RESURSI
Broj slajs registara	71 (1%)
Broj slajs LUT-ova	57 (0%)
Broj potpuno iskorišćenih parova LUT-FF	135 (1%)
Broj globalnih taktova	1 (4%)
Broj pinova	23 (34%)

Prilikom implementacije je korišćen samo jedan globalni takt, *clk*. Može se primetiti da dizajn nije utrošio mnogo resursa ploče, što je bilo i očekivano, s obzirom da se ne radi o komplikovanom interfejsu.

4.2. Verifikacija dizajna

Za potrebe verifikacije celokupnog dizajna, napisan je testbenč (*testbench*) Verilog modul. Testbenč je modul za koji se ne definišu portovi, samo se vrši instanciranje dizajna koji je potrebno testirati. U testbenču se generišu stimulusi ulaznih portova, za potrebe simulacije. Pre početka simulacije je potrebno definisati parametre *CLK_PERIOD* i *CLKS_PER_BIT*. Prvi parameter je određen frekvencijom *clk* signala ploče, dok je drugi količnik frekvencije *clk* signala i *baud rate* frekvencije, kao što je već objašnjeno.

U nastavku je prikazan kod kojim se deklarišu ulazni i izlazni parametri testebenča, a zatim instancira top-level dizajna odnosno DUT (*Device Under Test*):

```
`include "RS232_top_module.v"
```

```
  module RS232_tb ();

parameter CLK_PERIOD  = 40;
parameter CLKS_PER_BIT = 434;

//Ulazi
reg Clk = 0;
reg Tx_start = 0;
wire Tx_active;
wire Rx_line;
reg [7:0] Tx_data;
//Izlazi
wire Tx_done;
wire Rx_done;
wire Tx_line;
wire[7:0] Rx_data;
reg [2:0] State    = 3'b000;
reg [11:0] Clk_cnt = 0;
reg [2:0] Bit_index = 0;
reg [7:0] Rx_byte  = 0;

//Instanciranje dizajna
 RS232_top_module #(.CLKS_PER_BIT(CLKS_PER_BIT)) DUT
 (.clk(Clk),
  .top_Rx_line(Rx_line),
 .Tx_start(Tx_start),
  .Tx_data(Tx_data),
 .Rx_done(Rx_done),
 .Rx_data(Rx_data),
  .top_Tx_line(Tx_line),
 .Tx_done(Tx_done),
 .Tx_active(Tx_active)
  );
```

Testiran je primer *loopback* komunikacije, u cilju verifikacije i prijemnog i predajnog dela dizajna. U ovom test scenariju, 8-bitni podaci se šalju od predajnika *Tx* ka prijemniku *Rx* preko samo jedne žice, stoga je neophodno izvršiti sledeću dodelu:

```
assign Rx_line = Tx_line;
```

Generiše se takt signal, a zatim i stimulus – asertuje se signal Tx_start i tako započinje prenos podataka. Takođe, zadaju se podaci za prenos i smeštaju se u promenljivu Tx_data. Kraj simulacije određuje signal *Rx_done*, koji označava da su podaci uspešno primljeni.

Simulacija se realizuje na osnovu sledećih linija koda:

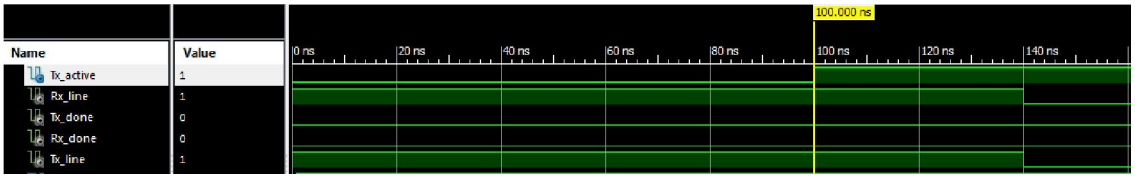
```
//Generisanje clock signala
always begin
 #(CLK_PERIOD/2) Clk <= !Clk;
end

initial begin   //Test1: slanje i prijem dva bajta podatka
 #(CLK_PERIOD);
 @(posedge Clk)
 begin
    Tx_start <= 1;
    Tx_data <= 8'hAB; // Prvi bajt podataka
    @(posedge Rx_done);
    @(posedge Rx_done);
    Tx_data <= 8'hCD; //Drugi bajt podataka
     @(negedge Rx_done) $finish;
  end
end

endmodule
```

U nastavku će biti date slike koje prikazuju sve faze simuliranja dizajna. Napomena: Sve naredne slike su dobijene opcijom Print Screen nakon pokretanja ISim simulatora.

Na prvom delu slike 4.2.1 je prikazan sam početak simulacije. To je trenutak kada se u testbenču signal *Tx_start* postavi na logički nivo 1. Promena se dešava za uzlaznu ivicu *clk* signala.

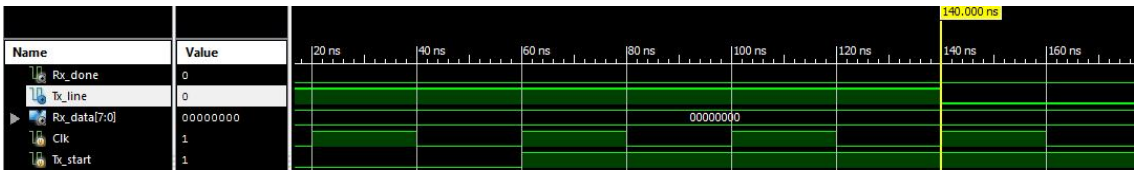
Slika 4.2.1 Prikaz ISim prozora nakon pokrenute simulacije – prvi deo

Nakon jednog vremenskog perioda *CLK_PERIOD* (40ns), *Tx_active* signal se asertuje:



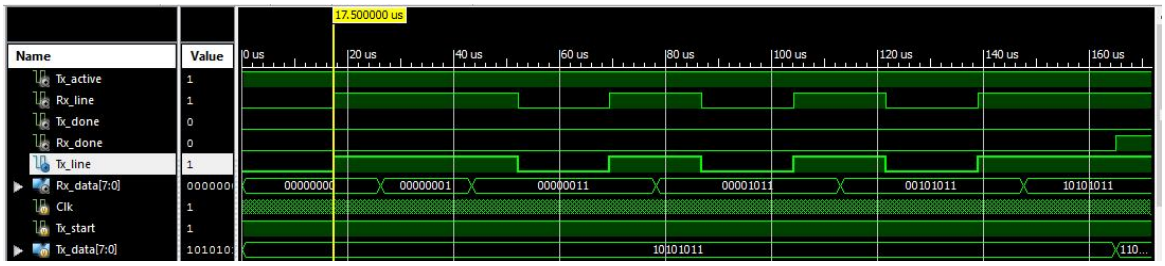
Slika 4.2.2 Prikaz ISim prozora nakon pokrenute simulacije – drugi deo

Nakon još jednog *clk* perioda *Tx_line* iz neaktivnog stanja prelazi u aktivno i započinje prenos slanjem start bita.



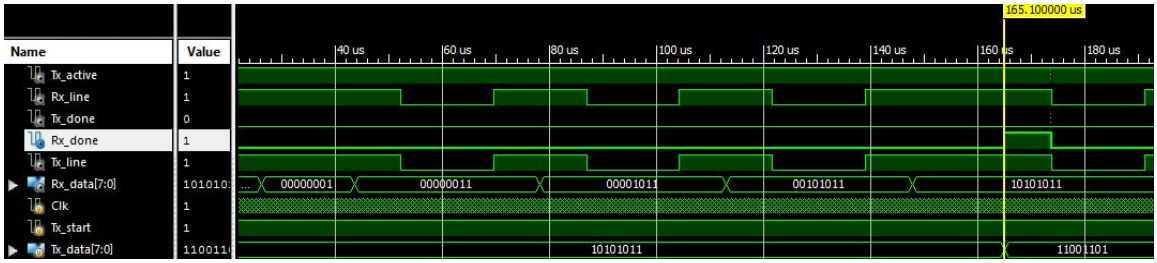
Slika 4.2.3 Prikaz ISim prozora nakon pokrenute simulacije – treći deo

Trenutak kada započinje prenos prvog bita podataka je prikazan na sledećoj slici 4.2.6, kao i celokupan prenos prvog paketa. Može se uočiti da se prvo prenosi LSB bit, kao i da promene *Tx_line* linije prate promene linije *Rx_line*, što je posledica *assign* dodele.



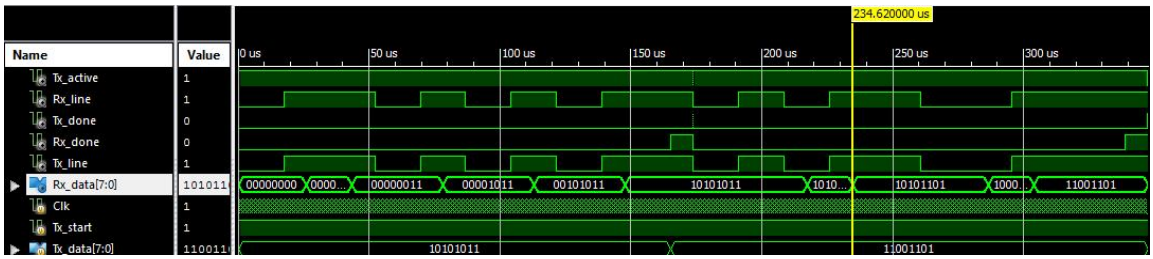
Slika 4.2.4 Prikaz ISim prozora nakon pokenute simulacije – četvrti deo

Slika 4.2.5 prikazuje kraj prenosa i asertovanje *Rx_done* signala, dok na *Tx_line* liniji traje prenos stop bita. Zatim počinje prenos start bita za drugi okvir podataka.



Slika 4.2.5 Prikaz ISim prozora nakon pokenute simulacije – peti deo

Na slici 4.2.6 je prikazan uspešan prenos i drugog paketa podataka, čime je simulacija završena. Potvrđeno je da dizajn radi prema RS-232 standardu.



Slika 4.2.6 Prikaz ISim prozora nakon pokenute simulacije – šesti deo

5. ZAKLJUČAK

U ovom radu su prikazane osnovne karakteristike RS-232 interfejsa i njegova hardverska realizacija. Uprkos razvoju tehnologije i standardizaciji novih protokola, RS-232 standard zbog svoje jednostavnosti i dalje ima veoma široku primenu.

Proračun performansi dizajna za FPGA platformu to i potvrđuje. Sam dizajn nije zauzeo mnogo resursa čipa, što govori u prilog jednostavnosti standarda. Kroz simulaciju je testirana komunikacija između realizovanih komponenti. Prenos podataka između predajnika i prijemnika je ušešno ostvaren.

Implementaciju algoritma je moguće unaprediti dodavanjem *parity* bita u okvir podataka, u cilju detekcije greške. Na taj način bi se dodatno poboljšao kvalitet prenosa. Kao što je rečeno u uvodu, RS-232 interfejs se može koristiti za povezivanje računara i FPGA ploče. Sa računara se mogu slati komande, što ostavlja prostor za razvoj biblioteke komandi. Ovaj rad bi mogao da predstavlja osnovu takvog projekta. Takođe, moguća je primena i u nastavne svrhe, kao primer dizajna konkretnog komunikacionog protokola.

LITERATURA

[1] Industrijski sistemi i protokoli – [Online]. Preuzeto sa :

http://www.keep.ftn.uns.ac.rs/predmeti/ee2_3g_indsys_protokoli/ISIP%20skripta%20%20Asinhroni%20prenos%20RS232%20i%20RS485.pdf

[2] Wikipedia, RS-232 – [Online]. Preuzeto sa:

https://en.wikipedia.org/wiki/RS-232

[3] RS232: Basics, Implementation & Specification – [Online]. Preuzeto sa:

http://www.engineersgarage.com/articles/what-is-rs232?page=3

[4] Zoran Čiča, Skripte sa predmeta Programiranje komunikacionog hardvera

[5] Onlajn simulator Adresa: https://www.edaplayground.com/