UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA TELEKOMUNIKACIJE

Projektna dokumentacija

ARHITEKTURE PAKETSKIH ČVORIŠTA

Realizacija VHDL testnog okruženja za 10G Ethernet baziranog na PCAP snimcima saobraćaja

Studenti:
Habib Sarajlić
Elmin Sejdić
Selma Smajlović
Nejla Šačić
Anes Uglješa

Sarajevo, januar 2020.

Sadržaj

Popis slika

Postavka projektnog zadatka

Naziv teme:

Realizacija VHDL testnog okruženja za 10G Ethernet baziranog na PCAP snimcima saobraćaja

Tematske zadaće:

1. Analizirati i teorijski obraditi format i sadržaj PCAP datoteka [1] sa posebnim osvrtom na primjere 10G Ethernet saobraćaja.
2. Testirati pomoću ModelSim-a Verilog implementaciju PCAP čitača/pisača sa Avalon-ST sučeljima datu na [2].
3. Nakon provedene simulacije analizirati signale na Avalon-ST sučeljima i uporediti ih sa očekivanim signalima opisanim u [3]. Ukoliko su potrebne korekcije, uvažiti ih pri realizaciji sljedećih zadataka.
4. Realizirati i testirati VHDL čitač PCAP datoteka sa Avalon-ST sučeljem na izlazu. Pogledati implementaciju datu na [4].
5. Realizirati i testirati VHDL pisač PCAP datoteka sa Avalon-ST sučeljem na ulazu.

Literatura:

1. https://wiki.wireshark.org/Development/LibpcapFileFormat
2. https://github.com/shuckc/verilog-utils/tree/master/pcap
3. 10-Gbps Ethernet MAC MegaCore Function, User Guide
4. https://github.com/ti-hsu-hh-de/VHDL-PCAP.

Poglavlje 1

Analiza strukture PCAP datoteka

1.1 PCAP datoteke

PCAP datoteke predstavljaju datoteke podataka kreirane pomoću programa i sadrže paketne podatke mreže. Ove datoteke se uglavnom koriste u analizi mrežnih karakteristika određenih podataka. PCAP datoteke se koriste za određivanje statusa mreže, omogućavajući analizatorima otkrivanje problema koji su se pojavili na mreži i proučavanje podatkovne komunikacije pomoću Wireshark-a. Kod unix sistema, i njima sličnih, pcap je implementiran u biblioteku libpcap.

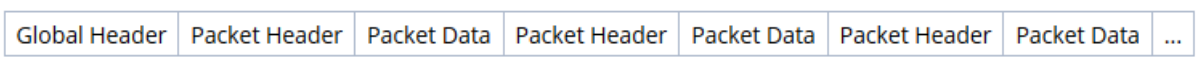
1.2 Format datoteke Libpcap

Libpcap predstavlja glavni format datoteke za snimanje koji se koristi u TcpDump/WinDump, snort i mnogim drugim alatima za umrežavanje. U potpunosti ga podržavaju Wireshark/TShark, koji međutim, sada po default-u generišu **pcapng** datoteke. Libpcap, koristi isti format datoteke kao i **WinPcap**. Iako se ponekad pretpostavlja da je ovaj format datoteke pogodan samo za **Ethernet** mreže, može posluživati različite tipove mreža.

Postoji nekoliko varijanti formata u praksi. Najčešće korišteni format je u verziji 2.4, i bit će objašnjen u nastavku.

Zvanična varijanta datoteke je verzija koja podržava vremenske oznake s preciznošću nanosekunde.

Format PCAP datoteke je definisan kao standard za snimanje paketa mrežnih podataka. Podaci su organizirani u binarnu datoteku koja se sastoji od globalnog zaglavlja (*Global Header*), nakon čega slijede paketi podataka sa individualnim zaglavljima (*Packet data* i *Packet Header*).



Slika 1.1: Format datoteke

Paket u snimljenoj datoteci ne mora da sadrži sve podatke koje sadrži paket na mreži. Snimljena datoteka može sadržavati najviše prvih N bajta svakog paketa. Vrijednost N predstavlja

"dužina snimke" (*snaplen*). Maksimalna vrijednost za N uzima se da je 65535, da bi se osiguralo da neće doći do odsijecanja paketa prilikom snimanja.

1.2.1 Global Header

Ovim zaglavljem počinje pcap datoteka, nakon kojeg slijedi prvo zaglavlje paketa.

```
typedef struct pcap_hdr_s {
        guint32 magic_number;   /* magic number */
        guint16 version_major;  /* major version number */
        guint16 version_minor;  /* minor version number */
        gint32  thiszone;       /* GMT to local correction */
        guint32 sigfigs;        /* accuracy of timestamps */
        guint32 snaplen;        /* max length of captured packets, in octets */
        guint32 network;        /* data link type */
} pcap_hdr_t;
```

Slika 1.2: Izgled Global Header-a

Sa slike se vidi da se globalno zaglavlje sastoji od sljedećih dijelova:

- **magic_number**: koristi se za otkrivanje samog formata datoteke i organizacije bajta. U ovo polje, aplikacija za pisanje zapisuje 0xa1b2c3d4 s izvornim formatom za organizaciju bajta. Aplikacija za čitanje očitat će ili 0xa1b2c3d4 (identično) ili 0xd4c3b2a1 (zamijenjeno). Ako aplikacija za čitanje pročita zamijenjenu vrijednost 0xd4c3b2a1, zna da će sva sljedeća polja također morati biti zamijenjena. Za datoteke s rezolucijom nanosekunde, aplikacija za pisanje piše 0xa1b23c4d, pri čemu su dvije nijanse dva bajta nižeg reda zamijenjene, a aplikacija za čitanje će očitati ili 0xa1b23c4d (identično) ili 0x4d3cb2a1 (zamijenjeno).
- **version_major, version_minor**: broj verzije formata datoteke (trenutna verzija je 2.4).
- **thiszone**: vrijeme korekcije u sekundama između GMT (UTC) i lokalne vremenske zone sljedećih vremenskih oznaka zaglavlja paketa. Primjeri: Ako su vremenske oznake u GMT (UTC), ovo polje je 0. Ako su vremenske oznake u srednjoevropskom vremenu (Amsterdam, Berlin, ...) koje je GMT + 1:00, vrijednost polja mora biti -3600. U praksi su vremenske oznake uvijek u GMT-u, tako da je vrijednost polja uvijek 0.
- **sigfigs**: teoretski, tačnost vremenskih oznaka u snimanju; u praksi postavljeno na 0.
- **snaplen**: "dužina snimke" za snimanje (obično 65535 ili više, ali može biti ograničen od strane korisnika).
- **network**: vrsta zaglavlja sloja veze, određuje vrste zaglavlja na početku paketa (npr. za Ethernet ima vrijednost 1).

1.2.2 Record (Packet) Header

Svaki snimljeni paket sadrži zaglavlje paketa.

```
typedef struct pcaprec_hdr_s {
        guint32 ts_sec;         /* timestamp seconds */
        guint32 ts_usec;        /* timestamp microseconds */
        guint32 incl_len;       /* number of octets of packet saved in file */
        guint32 orig_len;       /* actual length of packet */
} pcaprec_hdr_t;
```

Slika 1.3: Packet header

- **ts_sec** : vrijeme i datum kada je paket snimljen. Ova vrijednost je u sekundama od 1. januara 1970. 00:00:00 GMT.
- **ts_usec**: u standardnim pcap datotekama ova vrijednost označava broj mikrosekundi kada je paket snimljen, kao odstupanje od *ts_sec*. U datotekama rezolucije nanosekunde, ova vrijednost označava broj nanosekundi kada je paket snimljen, kao odstupanje od *ts_sec*.
- **incl_len**: broj bajta podataka paketa koji su ispravno snimljeni i sačuvani u datoteci. Ova vrijednost nikada ne smije postati veća od *orig_len* ili snaplen vrijednosti globalnog zaglavlja.
- **orig_le**: dužina paketa na mreži. Ukoliko se *incl_len* i *orig_len* razlikuju, spremljena veličina paketa bila je ograničena snaplen-om.

1.2.3 Packet Data

Stvarni podaci paketa odmah će uslijediti nakon zaglavlja paketa kao blok podataka *incl_len* bajta bez određenog poravnanja bajta.

Poglavlje 2

Avalon ST interfejs sa 10G Ethernetom

2.1 Avalon ST interfejs

Avalon Streaming (Avalon-ST) interfejs koristi se za komponente koje pokreću visoku propusnost, nisku latenciju, jednosmjerne podatke. Tipične aplikacije uključuju multipleksirane tokove, pakete i DSP podatke. Signali Avalon-ST interfejsa mogu opisati tradicionalna sučelja za streaming koji podržavaju jedan tok podataka bez znanja o kanalima ili granicama paketa. Interfejs također može podržati složenije protokole koji imaju mogućnost burst-a i prijenosa paketa s paketima koji su prepleteni preko više kanala.
Avalon-ST sučelja podržavaju podatkovne putanje koje zahtijevaju sljedeće karakteristike:

- Nisku latenciju, visoka propusnost u *point-to-point* prijenosu podataka
- Podršku za više kanala s fleksibilnim interleaving-om paketa
- Sideband signalizaciju o kanalu, grešci, te početnom i završnom razgraničenju paketa
- Podršku za obradu podataka
- Automatsku adaptaciju interfejsa

2.1.1 Signali Avalon ST interfejsa

Osnovni signali:
channel - broj kanala za prenos podataka za trenutni ciklus. Ako interfejs podržava *channel* signal, također mora definirati i *maxChannel* parametar (maksimalni broj kanala koje podatkovni interfejs može podržati).
data - signal podataka od source-a do sink-a, obično nosi većinu informacija koje se prenose.
error - bit maska za označavanje grešaka koje utječu na podatke koji se prenose u trenutnom ciklusu. Jedan jedini bit signala greške maskira svaku od grešaka koju komponenta prepoznaje.
ready - signal može biti 0 ili 1. Ukoliko je 1, označava da sink može prihvatiti podatke.
valid - source koristi ovaj signal kako bi kvalificirao sve ostale source-sink signale.
Signali prenosa paketa:

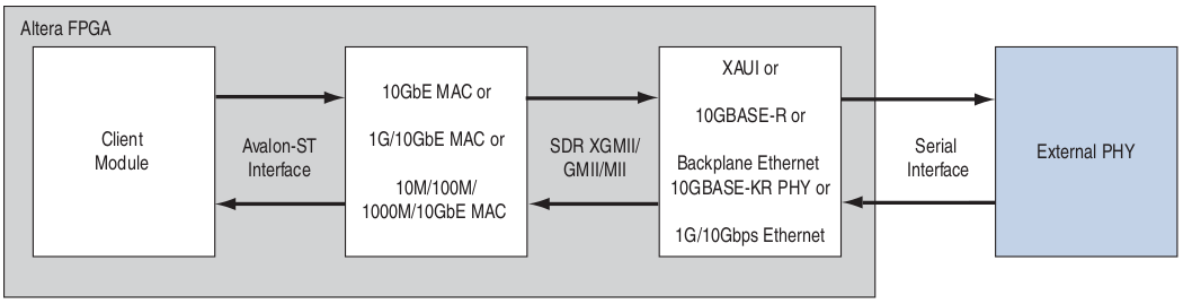
empty - označava broj praznih simbola, odnosno, broj simbola koji ne predstavljaju valjane podatke.
startofpacket- označava početak paketa.
endofpacket - označava kraj paketa.

2.2 10G Ethernet

10-Gbps Ethernet(10GbE) Media Access Controller (MAC) IP jezgra je konfigurabilna komponenta koja implementira IEEE 802.3-2008 specifikaciju. IP jezgra nudi sljedeće modove:

- **10 Gbps mod** - koristi Avalon Streaming (Avalon-ST) interfejs na klijentskoj strani i pojedinačnu brzinu (SDR) XGMII na strani mreže.
- **1 Gbps/10 Gbps mod** - koristi Avalon-ST interfejs na klijentskoj strani i GMII/SDR XGMII na strani mreže
- **10 Mbps/100 Mbps/1 Gbps/10 Gbps (10M-10G) mod** - koristi Avalon-ST interfejs na klijentskoj strani i MII/GMII/SDR XGMII na strani mreže.

Za izgradnju kompletnog Ethernet podsistema u Altera uređajima i konekciju sa eksternim uređajima, može se koristiti 10GbE MAC IP jezgro sa Altera PHY IP jezgrom, kao i sa XAUI PHY u FPGA čvrsto silikonsko-integrisanom u XAUI PHY, 10GBASE-R PHY, Backplane Ethernet 10GBASE-KR PHY, ili 1G/10Gbps Ethernet PHY IP



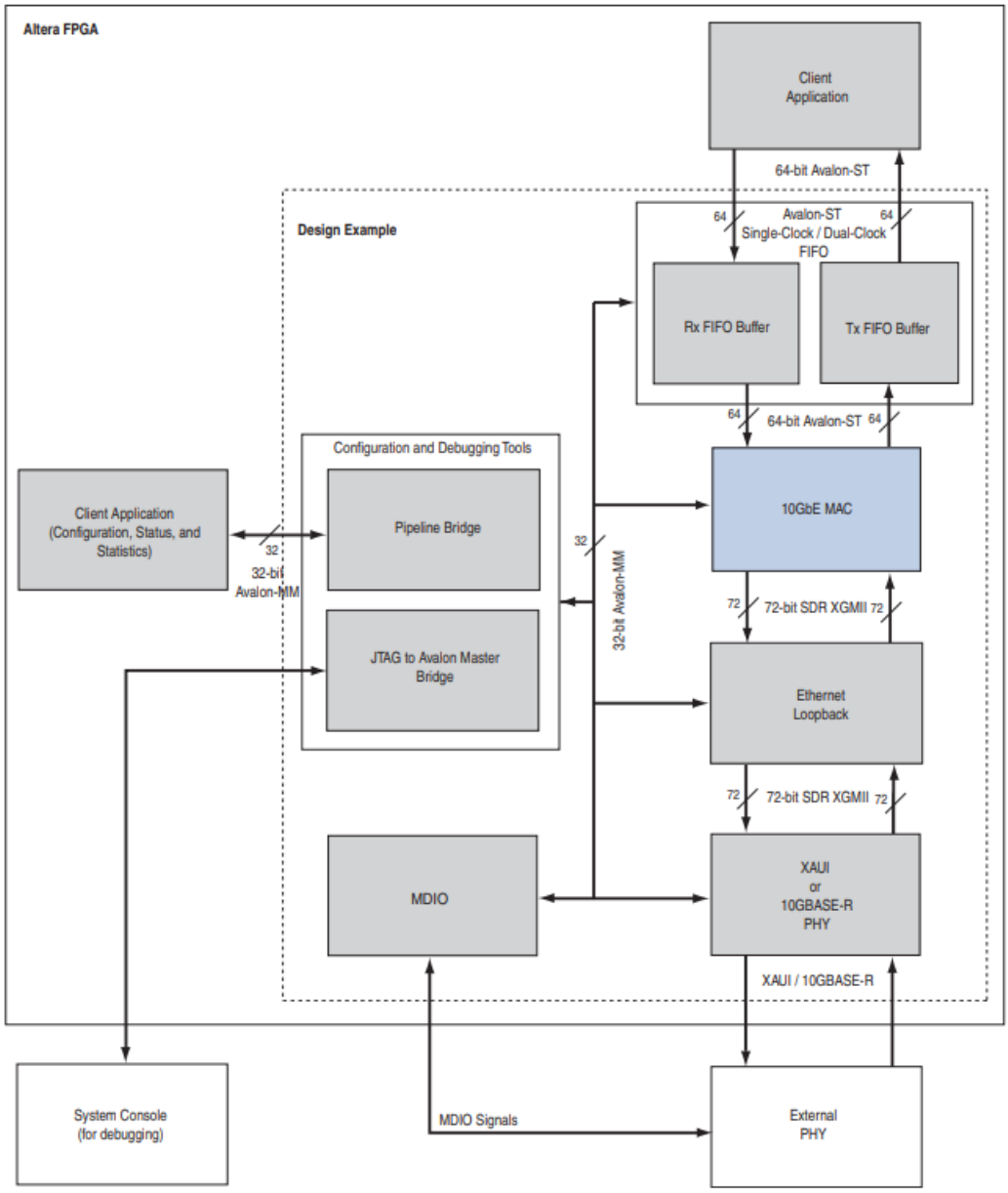
Slika 2.1: Primjena 10GbE MAC

2.2.1 Osobine

10GbE MAC ima sljedeće osobine:

- Režim rada: 10 Mbps, 100 Mbps, 1 Gbps i 10 Gbps;
- Avalon-ST 64-bitni klijent interfejs na 156.25 MHz;
- Direktni interfejs za 4-bitni MII 125 MHz sa clock signalom; 2.5 MHz za 10 Mbps i 25 MHz za 100 Mbps;
- Direktni interfejs za 8-bitni GMII na 125 MHz;
- Direktni interfejs za 64-bitni SDR GMII na 156.25 MHz;
- Mogućnost programiranja podatkovne putanje za omogućavanje odvojenog instanciranja MAC TX bloka, MAC RX bloka, ili oba bloka MAC TX i MAC RX.

2.2.2 10GbE primjeri dizajna komponenti



Slika 2.2: 10GbE primjer dizajna blok šeme

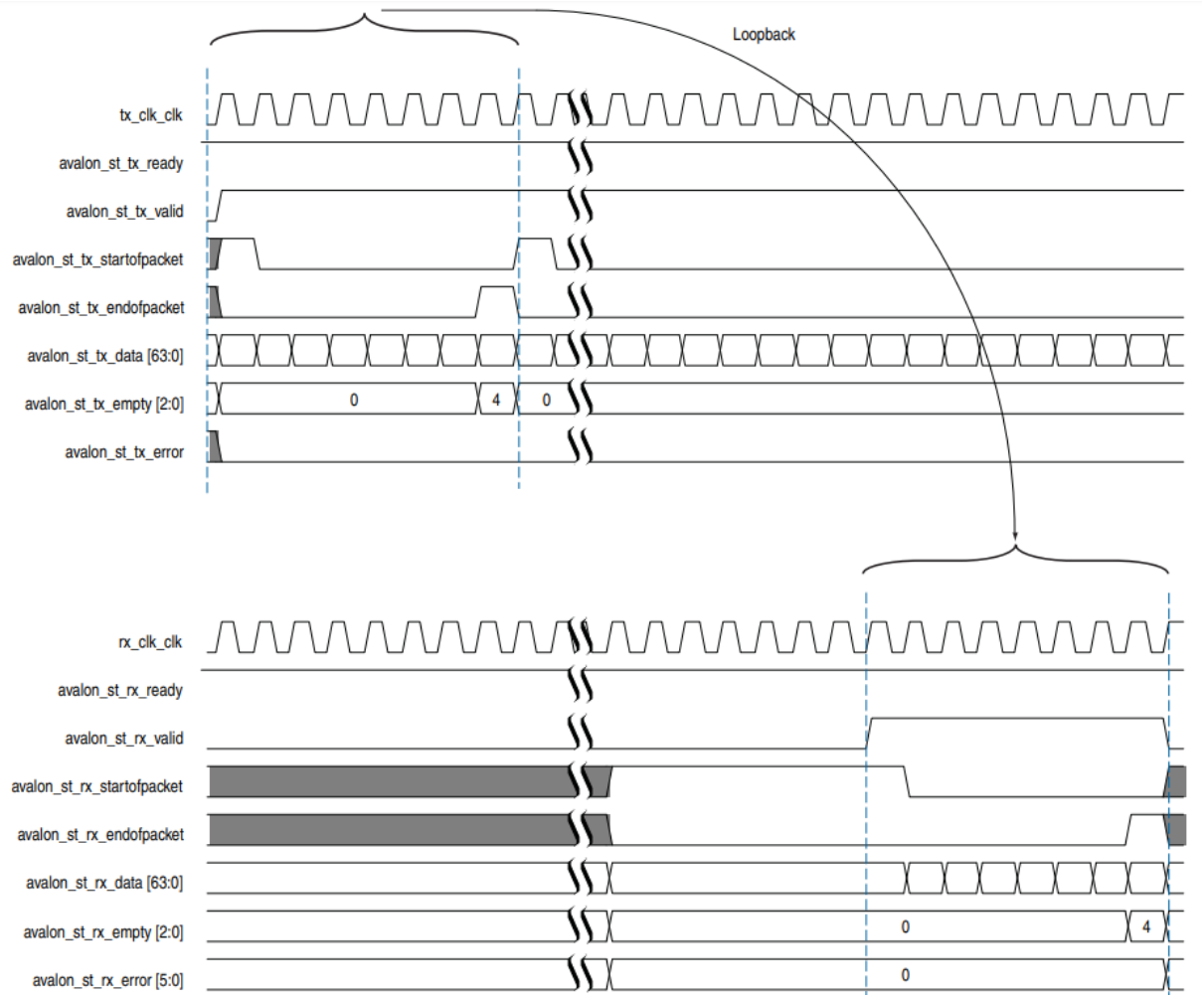
Ovaj primjer dizajna uključuje sljedeće komponente:

- **10GbE Ethernet MAC** - MAC IP jezgra sa zadanim postavkama. Ova IP jezgra uključuje memorijski-bazirane statističke brojače
- **XAUI PHY or 10GBASE-R PHY** - PHY IP jezgra sa zadanim postavkama. XAUI PHY je podešen na **Hard Xaui**.
- **Ethernet Loopback** - modul petlje pruža mehanizam za verifikaciu funkcionalnosti MAC and PHY.

- **RX and TX FIFO buffers** - Avalon-ST Single-Clock ili Dual-Clock FIFO jezgre preko kojih baferi primaju i šalju podatke između MAC i klijenta. Ovi FIFO baferi su 64 bita široki i 512 bita duboki. Zadana konfiguracija je Avalon-ST Single-Clock FIFO, koja radi u skladišti i proslijedi modu i može se konfigurisati da pruža paketski-bazirano izbacivanje kada se dogodi geška.
- **Configuration and debugging tools** - pruža pristup registrima sljdećih komponenata sa Avalon Memory-Maped (Avalon-MM) interfejs: MAC, MDIO, Ethernet Loopback, PHY i FIFO baferi. Pruža testiranje uključujući Avalon drajver koji koristi pipeline bridge za pristup registrima. Može se koristiti sistemska konzola za pristup registrima sa JTAG do Avalon Master Bridge jezgre koja verifikuje dizajn u hardver.

2.2.3 10GbE simulacijski vremenski dijagrami

Slika 2.3 ilustruje detaljan opis prijenosa i prijema paketa.



Slika 2.3: Prijenos i prijem paketa

Funkcionalnosti Avalon-ST signala:
avalon_st_tx_startofpacket - signal koji indicira početak paketskog prenosa, tj. signal je aktivan u prvom ciklusu transfera paketa.
avalon_st_tx_endofpacket -signal koji indicira kraj paketskog prenosa, tj. signal je aktivan u posljednjem ciklusu transfera paketa.
avalon_st_tx_valid - signal koji indicira da je podatak koji se prenosi u trenutnom ciklusu transfera paketa ispravan.
avalon_st_tx_ready - signal koji indicira da je odredište spremno za prelazak u sljedeći ciklus transfera paketa.
avalon_st_tx_data - podaci.
avalon_st_tx_empty - indicira broj praznih simbola u podatku iz trenutnog ciklusa transfera paketa, a najčešće se koristi u posljednjem ciklusu transfera paketa.
avalon_st_tx_error - signal koji indicira da trenutni paket sadrži greške.

Poglavlje 3

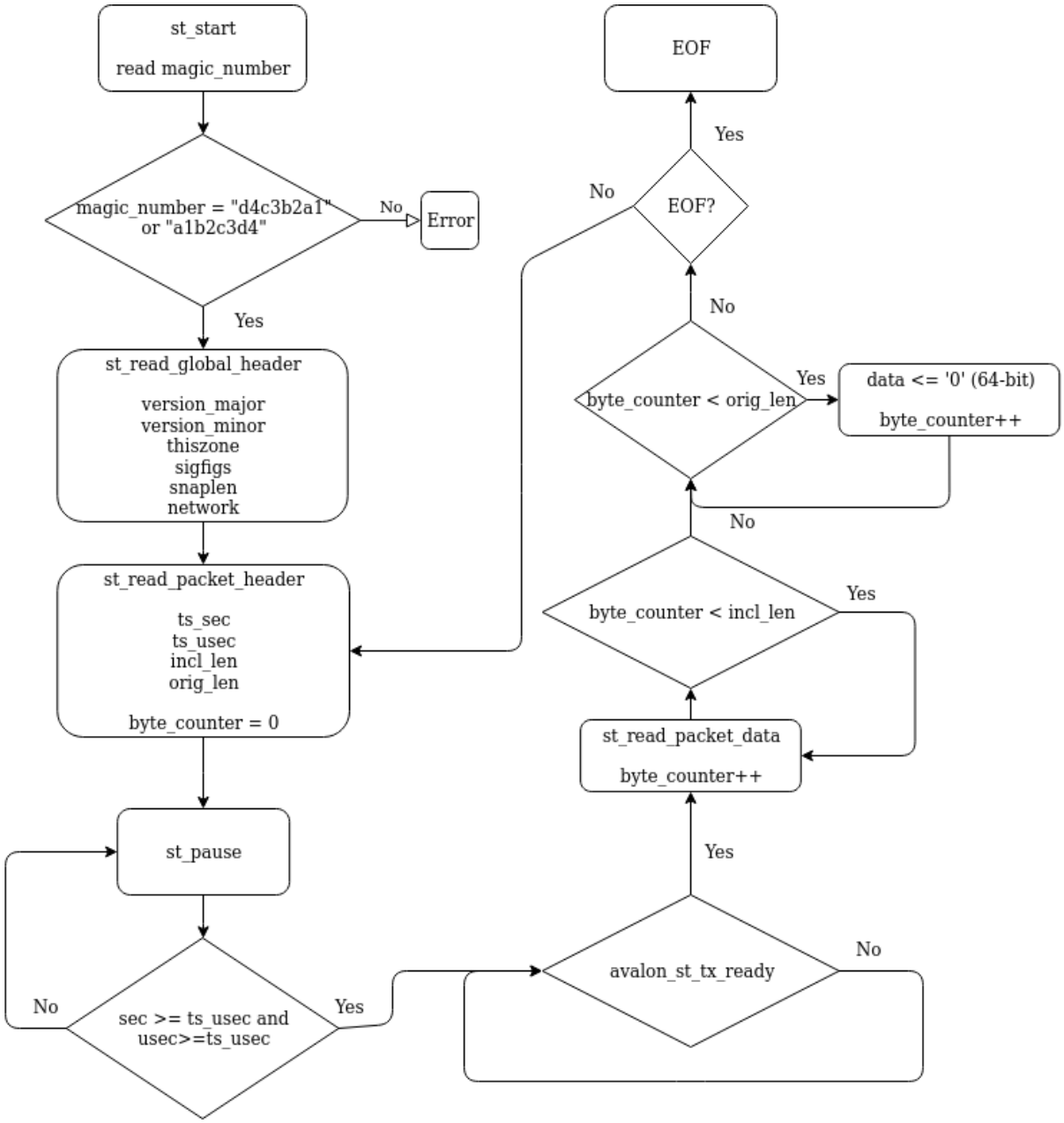
Čitač i pisač PCAP datoteka u VHDL-u

Analizom signala na *Analon ST* interfejsu uočeno je da postoje 2 osnovna stanja u kojima se može naći Avalon ST interfejs. Prvo stanje je stanje pauze u kojem interfejs čeka *startofpacket* signal, a drugo stanje je stanje nakon dolaska *startofpacket* signala, odnosno, stanje u kojem interfejs prima podatke sve do dolaska *endofpacket* signala nakon kojeg interfejs prelazi u prvo stanje. Ovakav način rada *Analon ST* interface-a moraju implementirati i čitač i pisač PCAP datoteka kako bi bili kompatibilni *Avalon ST* interfejsom. Kako je riječ o 10Gb Ethernetu širina sabirnice kojom se podaci prenose je 64 bita i frekvencija clock signala je 156.25 MHz.

3.1 Čitač PCAP datoteka

Čitač PCAP datoteka treba da bude kompatibilan sa *Avalon ST* interfejsom, odnosno da na svom izlazu ima *Avalon ST source* interfejs, te da može generisati pakete brzinom 10 Gbps. U kodu ispod je prikazan *entity* čitača PCAP datoteka. Možemo vidjeti da čitač PCAP datoteka ima sve signale kao i *Avalon ST source* interfejs.

```
entity pcap_reader is
  port (
    clk                         : in  std_logic;--! system clock
    reset                       : in  std_logic;
    avalon_st_tx_ready          : in  std_logic;
    avalon_st_tx_valid          : out std_logic;
    avalon_st_tx_startofpacket  : out std_logic;
    avalon_st_tx_endofpacket    : out std_logic;
    avalon_st_tx_data           : out std_logic_vector(63 downto 0);
    avalon_st_tx_s_empty        : out std_logic_vector(2 downto 0);
    avalon_st_tx_error          : out std_logic
    );
end pcap_reader;
```



Slika 3.1: Flow dijagram čitača PCAP datoteka

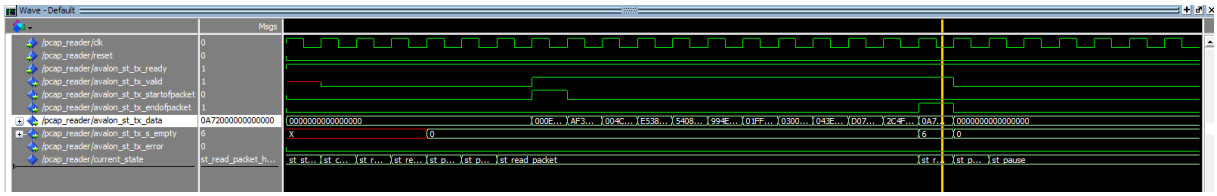
PCAP čitač prvo čita *magic_number* iz globalnog zaglavlja na osnovu kojeg detektuje da li je pcap datoteka zapisana u *big* ili *little endian* formatu. Signali označeni sa *s_ime_signala*, kao npr. *s_version_major, s_thiszone, s_ts_sec, s_orig_len, itd.* su "raw" signali, odnosno, signali koji su direktno pročitani iz datoteke tj. bez konverzije iz *big* u *little endian* i obrnuto. Signali sa prefiksom sd tj. signali označeni kao *sd_ime_signala*, kao npr. *sd_version_major, sd_thiszone, sd_ts_sec, sd_orig_len, itd.* su signali koji su konvertovani u *big endian* ukoliko je *magic_number* "d4c3b2a1". Ova implementacija PCAP čitača koristi *big endian* signale.

Nakon iščitavanja *global headera* čitač prelazi u stanje *st_read_packet_header* u kojem čita zaglavlje paketa. Nakon čitanja zaglavlja paketa, čitač PCAP datoteka prelazi u stanje pauze te čeka da lokalni tajmer primi vrijednosti sekundi i mikrosekundi koje su pročitane iz zaglavlja paketa, nakon čega prelazi u stanje *st_read_packet_data*. U zaglavlju paketa se pored vremena emisije paketa nalaze i parametri *incl_len* i *orig_len*. Značenje ovih parametara je objašnjeno u prethodnom poglavlju, međutim, važno je napomenuti da oni ne moraju biti jednaki te da čitač

čita iz fajla broj bajta koji određuje parametar *incl_len*. Ukoliko parametri *incl_len* i *orig_len* nisu jednaki, tada čitač PCAP fajlova na izlaz šalje nule kako bi ispoštovao originalnu veličinu paketa.

Kako je *data* signal 64bitna sabirnica, tj. na jednu ivicu clock signala se šalje na izlazni interfejs 8 bajta, moguć scenario je da dužina paketa koji se šalje na izlazni interfejs nije djeljiva sa 8. U takvom scenariju, kada se posljednji bajti tog paketa šalju na izlaz (a njih je manje od 8) tada se *empty* signalom određuje koliko bajta je prazno od mogućih 8 bajta.

Čitač PCAP datoteka poštuje i *avalon_st_tx_ready* signal, odnosno, ne započinje slanje paketa dok je *avalon_st_tx_ready* na nivou logičke 0. Signali *avalon_st_tx_valid*, *avalon_st_tx_startofpaceket*, *avalon_st_tx_endofpacket* i *avalon_st_tx_error* su također ispravno upravljani što se vidi sa slike 3.2.

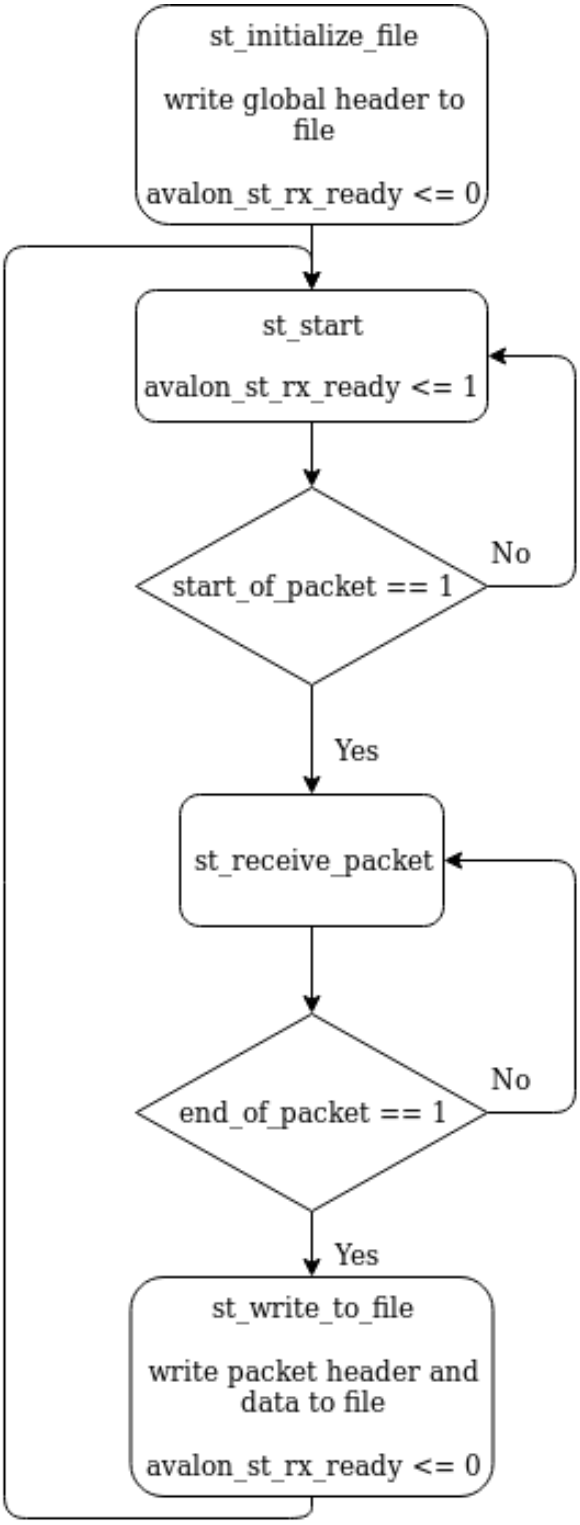


Slika 3.2: Prikaz signala na izlazu čitača PCAP datoteka

3.2 Pisač PCAP datoteka

Pisač PCAP datoteka također treba da bude kompatibilan sa *Avalon ST* interfejsom, odnosno da na svom ulazu ima *Avalon ST sink* interfejs, te da može primati pakete brzinom 10 Gbps. U kodu ispod je prikazan *entity* pisača PCAP datoteka. Možemo vidjeti da pisač PCAP datoteka ima sve signale kao i *Avalon ST sink* interfejs.

```
entity pcap_writer is
  port (
    clk                          : in  std_logic; --! system clock
    reset                        : in  std_logic;
    avalon_st_rx_ready           : out std_logic;
    avalon_st_rx_valid           : in  std_logic;
    avalon_st_rx_startofpacket   : in  std_logic;
    avalon_st_rx_endofpacket     : in  std_logic;
    avalon_st_rx_data            : in  std_logic_vector(63 downto 0);
    avalon_st_rx_empty           : in  std_logic_vector(2 downto 0);
    avalon_st_rx_error           : in  std_logic
    );
end pcap_writer;
```



Slika 3.3: Flow dijagram pisača PCAP datoteka

Pisač PCAP datoteka prvo ulazi u stanje *st_initialize_file* u kojem zapisuje *global header* u PCAP datoteku. U ovom stanju pisač nije u mogućnosti da prima pakete pa je signal *avalon_st_rx_ready* na logičkoj 0. Nakon zapisivanja globalnog zaglavlja pisač prelazi u stanje *st_start*. Sada je pisač spreman za primanje paketa pa je *avalon_st_rx_ready* na logičkoj 1. Pisač čeka signal *startofpacket* nakon kojeg prelazi u stanje *st_receive_packet*. U ovom stanju pisač PCAP datoteka primljene bajte sprema u RAM memoriju. Razlog spremanja paketa u

RAM, umjesto direktnog zapisivanja u datoteku jeste taj što se u PCAP datoteku prvo mora zapisati packet header pa tek onda packet data, što dalje zahtijeva poznavanje dužine paketa prije njegovog prijema. Kako čitač nema načina da sazna veličinu paketa, on paket sprema u RAM memoriju nakon čega prelazi u stanje *st_write_to_file*. Sada pisač PCAP datoteka zna dužinu paketa pa je u mogućnosti kreirati packet header. Nakon zapisivanja packet headera, pisač vrši zapis paketa iz RAM memorije u datoteku. U ovom stanju pisač nije u mogućnosti da prima naredni paket pa je *avalon_st_rx_ready* na logičkoj 0. Važno je napomenuti da se prvi dio zaglavlja paketa formira pri dolasku signala *startofpacket*, tj. vrši se postavljanje varijabli *s_ts_usec* i *s_ts_sec* na vrijednost lokalnog tajmera, dok se drugi dio zaglavlja formira nakon dolaska *endofpacket* signala.

3.3 Tajmer

Za potrebe preciznog mjerenja trenutka slanja paketa kod čitača PCAP datoteka, odnosno, trenutka dolaska paketa kod pisača PCAP datoteka razvijen je tajmer koji na svom izlazu daje vrijednost proteklog vremena sa preciznošću jedne mikro sekunde.

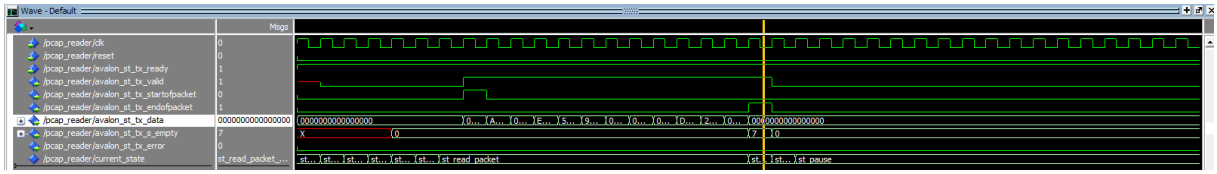
```
entity timer is
  port (
    clk    : in  std_logic;--! system clock
    reset  : in std_logic;
    u_sec  : out std_logic_vector (31 downto 0);
    second : out std_logic_vector (31 downto 0)
);
end timer;
```

Frekvencija clock signala je frekvencija koju koriste PCAP čitač i pisač te *Avalon ST* interfejs i iznosi 156.25 MHz. Trajanje jednog takt intervala u ovom slučaju je 6.4 ns, što predstavlja problem jer broj takt intervala koji je potreban da bi se napravila pauza od 1 μs nije cio broj i iznosi 156.25. Kako je moguće izbrojati samo cijeli broj takt intervala, tajmer broji 156 takt intervala te nakon izbrojenih 156 takt intervala povećava vrijednost brojača mikro sekundi. Kako bi se izbjegla akumulacija greške od 0.25 takt intervala, brojač svako 4. mikro sekundu preskoči brojanje 1 takt intervala i na taj način se postiže da je svaka 4. mikrosekunda tačna, odnosno, najveća greška koju tajmer napravi iznosi 6.4 ns.

3.4 Testiranje čitača i pisača PCAP datoteka

Za potrebe testiranja čitača PCAP datoteka kreirali smo "custom" PCAP datoteku u kojoj smo varijablu *orig_len* postavili:

1. da bude ista kao *incl_len*, tj. da se cijeli paket nalazi u PCAP datoteci (vidi sliku 3.2),
2. da bude veća od *incl_len*, tj. da dio paketa nije snimljen u PCAP datoteku, što znači da čitač PCAP datoteka mora vršiti dopunu paketa nulama (vidi sliku 3.4).



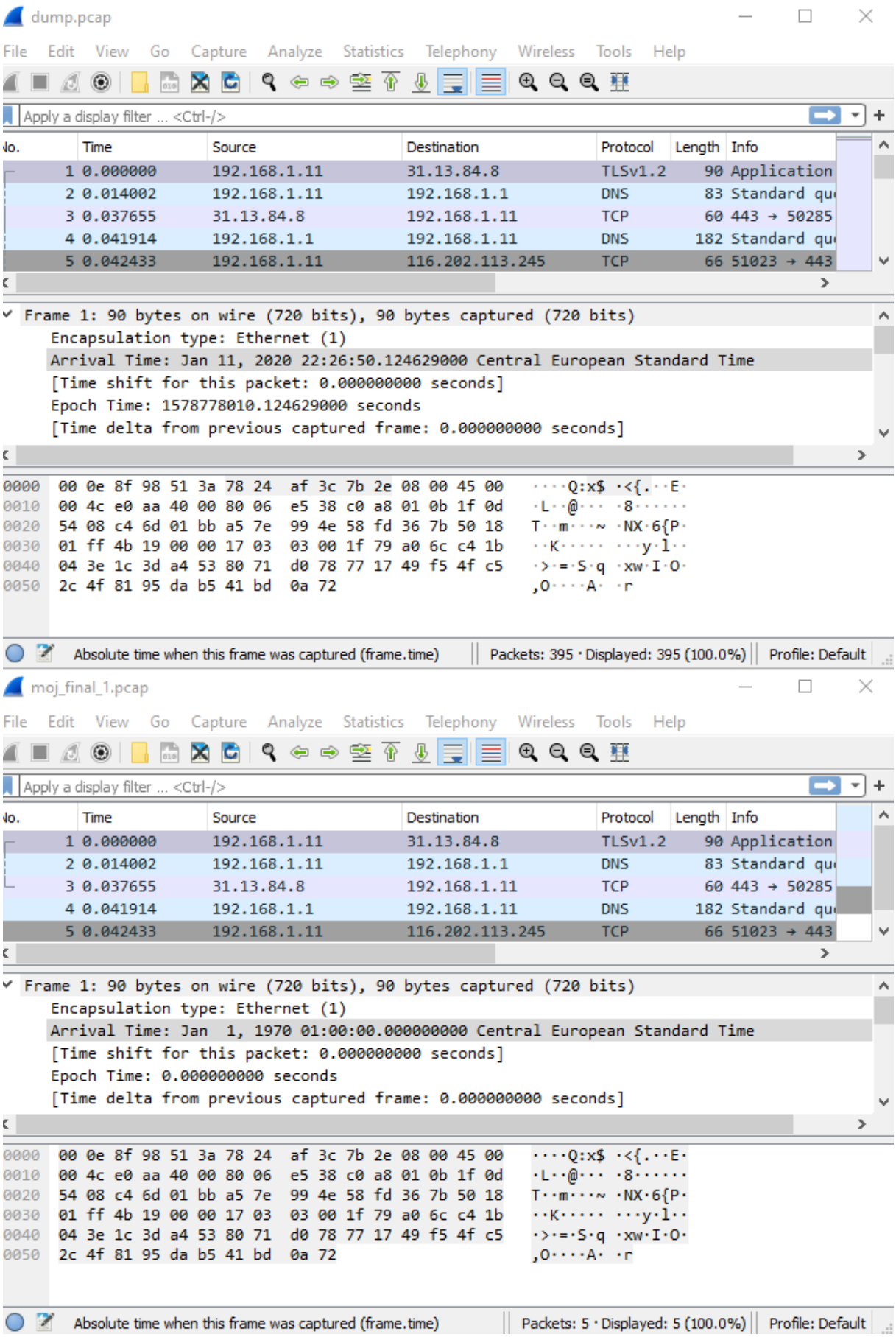
Slika 3.4: Prikaz signala na izlazu čitača PCAP datoteka kada se vrši dopuna paketa nulama

Pri snimanju slike 3.4 korišten je PCAP file kod kojeg u zaglavlju prvog paketa parametar *incl_len* ima vrijednost 90, dok parametar *orig_len* ima vrijednost 97. To znači da će čitač PCAP datoteka izvršiti dopunu paketa sa 7 bajta nula, odnosno, u pretposljednjem taktu će na izlaz biti poslana 2 (jer je 90B - 8B*11 = 2B) posljednja bajta iz PCAP fajla te 6 bajta popunjenih nulama. U posljednjem takt intervalu čitač PCAP datoteka će na izlaz poslati 8 bajta popunjenih nulama, od kojih je prvi bajt dio paketa, a ostalih 7 su prazni bajti, tako da je vrijednost *empty* signala 7.

Za testiranje pisača PCAP datoteka kreiran je *VHDL* fajl u kojem je izlaz čitača PCAP datoteka spojen na ulaz pisača PCAP datoteka. Sa ovakvom strukturom, datoteku koju pročita PCAP čitač treba da dobijemo u datoteci koju zapiše PCAP pisač.

```
entity pcap_reader_writer_test is
  port (
    clk    : in std_logic;--! system clock
    reset  : in std_logic
);
end pcap_reader_writer_test;
architecture arch of pcap_reader_writer_test is
  signal s_avalon_st_ready          : std_logic;
  signal s_avalon_st_valid          : std_logic;
  signal s_avalon_st_startofpacket  : std_logic;
  signal s_avalon_st_endofpacket    : std_logic;
  signal s_avalon_st_data           : std_logic_vector(63 downto 0);
  signal s_avalon_st_empty          : std_logic_vector(2 downto 0);
  signal s_avalon_st_error          : std_logic;
  component pcap_reader is
    port (...);
  end component;
  component pcap_writer is
    port (...);
  end component;
 begin
    veza_a: pcap_reader port map(clk, reset, s_avalon_st_ready, ...);
    veza_b: pcap_writer port map(clk, reset, s_avalon_st_ready, ...);
end arch;
```

Nakon pokretanja ovakve strukture (najduže što je Modelsim dozvolio je oko 100 ms) u izlaznoj datoteci vidimo identičan sadržaj kao u ulaznoj datoteci osim zaglavlja paketa. Razlika u zaglavlju paketa je ta što pisač PCAP datoteka nema načina da sazna trenutno vrijeme pa sve pakete snima kao da su pristigli 1970. godine (što odgovara *UNIX* vremenu kada varijabla koja predstavlja *UNIX* vrijeme (*ts_sec* i *ts_usec*) ima vrijednost blisku 0). Možemo primijetiti da je razmak između paketa ostao isti, što se vidi sa slike 3.5.



Slika 3.5: Sadržaj PCAP datoteke nakon simulacije PCAP pisača

Poglavlje 4

Zaključak

Tema ovog projektnog zadatka je bila realizacija VHDL testnog okruženja za 10G Ethernet baziranog na PCAP snimcima saobraćaja, što je i realizovano. Implementirani su i testirani čitač i pisač PCAP datoteka. Čitač i pisač moraju biti kompatibilni sa Avalon ST interfejsom i moraju biti u mogućnosti generisati odnosno primati pakete brzinom 10 Gbps.

Analiza PCAP datoteka te analiza načina rada signala *Avalon ST* interfejsa je najvažniji dio ovog projekta i to radi postizanja kompatibilnosti pisača i čitača sa ostalim uređajima koji koriste *Avalon ST* interfejs, te radi postizanja kompatibilnosti datoteke koju zapiše PCAP pisač sa ostalim software-ima koji podržavaju PCAP datoteke.

Druga bitna stvar u ovoj implementaciji je problem nastao zbog nemogućnosti dobijanja cijelog broja takt intervala. Naime, kako je frekvencija clocka signala koji koriste PCAP čitač, pisač te *Avalon ST* interfejs 156.25 MHz, a trajanje jednog takt intervala 6.4 ns, to predstavlja problem jer broj takt intervala koji je potreban da bi se napravila pauza od 1 μs nije cio broj tj. iznosi 156.25.

Testiranje čitača PCAP datoteka je bilo relativno jednostavno jer je bilo potrebno samo pokrenuti simulaciju i vizuelno pregledati signale da li prate *Avalon ST* standard i provjeriti da li podaci na signalu *data* odgovaraju podacima koji se nalaze u PCAP datoteci koja se iščitava. Testiranje pisača PCAP datoteka realizovano je tako što je izlaz čitača PCAP datoteka spojen na ulaz pisača. Nakon pokretanja takve strukture u *Modelsimu* izvršili smo analizu zapisane datoteke i poredili je sa datotekom iz koje je PCAP čitač čitao.

Kao jednostavan format datoteke, PCAP ima prednost što je kompatibilan s gotovo svim programima za praćenje paketa, s nizom verzija za Windows, Linux i Mac OS. Snimanje paketa može se primijeniti u gotovo bilo kojem okruženju.

Ovaj rad je rezultovao uspješno implementiranim i testiranim modelom čitača i pisača PCAP datoteka, ali može poslužiti npr., i kao osnova za daljnji rad na poboljšanju ukupne sigurnosti na mreži, testiranje raznih vrsta firewall-a i sigurnosnih protokola.