ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

IMPLEMENTACIJA PROVERE ISPRAVNOSTI IP ZAGLAVLJA I SEGMENTACIJE IP PAKETA ZA 10G PORTOVE RUTERA

–Master rad –

Kandidat:

Miloš Đokić 2013/3240

Mentor:

doc. dr Zoran Čiča

Beograd, Novembar 2014.

SADRŽAJ

1. UVOD

Četvrta verzija IP protokola, IPv4, i dalje predstavlja dominantnu verziju IP protokola na Internetu. IP protokol omogućava usmeravanje IP paketa kroz Internet mrežu, između polaznog i odredišnog mrežnog uređaja, na osnovu IP adrese u zaglavljima tih paketa. Usmeravanje paketa vrše ruteri koji predstavljaju najbitnije uređaje u Internet mreži. Kapacitet rutera se stalno povećava kako bi bili u stanju da isprate rastuće zahteve za protokom koje postavlja sve veći broj servisa na Internetu. Zbog toga ruteri postaju sve kompleksniji uređaji koji moraju biti u stanju da obrade veliki broj IP paketa u jedinici vremena.

Obrada jednog IP paketa u ruteru počinje njegovim izdvajanjem iz okvira (jedinica podataka koja se razmenjuje na sloju linka podataka). Zatim se vrši analiza ispravnosti zaglavlja tog paketa. Ako je zaglavlje ispravno, iz njega se izdvaja IP adresa odredišta i vrednosti ostalih bitnih polja, a paket se segmentira na ćelije jednake dužine. Na osnovu odredišne IP adrese i tabele rutiranja određuje se izlazni port rutera na koji će paket biti prosleđen. Na izlaznom portu od ćelija se ponovo sastavlja IP paket, koji se zatim enkapsulira u odgovarajući okvir.

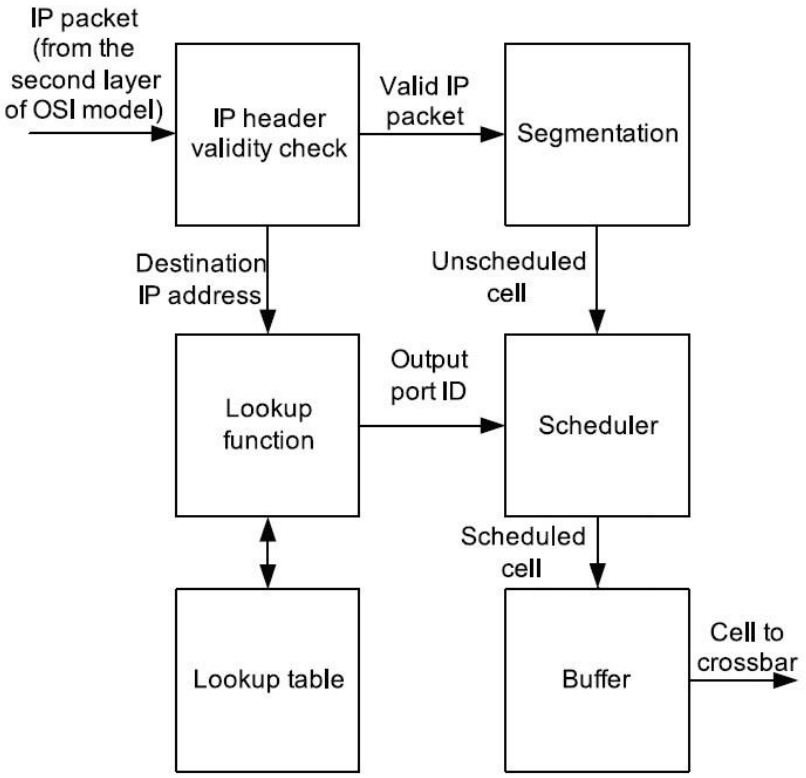
Ovaj rad se bavi hardverskom implementacijom provere ispravnosti IPv4 zaglavlja i segmentacije IP paketa na ćelije jednake dužine. Kako na savremenim 10G portovi postaju uobičajeni, predviđeno je da ova implementacija bude kompatibilna upravo sa takvim portovima. Za realizaciju implementacije korišćen je VHDL jezik, a razvoj i funkcionalna verifikacija dizajna sprovedeni su u ISE razvojnom okruženju za FPGA čipove proizvođača Xilinx.

Rad je organizovan na sledeći način. Drugo poglavlje sadrži kratak opis obrade primljenog paketa, objašnjenje strukture IPv4 zaglavlja, opis postupka provere ispravnosti IPv4 zaglavlja, kao i opis postupka segmentacije IP paketa. Treće poglavlje detaljno opisuje strukturu dizajna i VHDL kod napisan za potrebe hardverske implementacije dizajna. Četvrto poglavlje daje opis verifikacije rada dizajna poređenjem vrednosti različitih signala tokom funkcionalne simulacije sa njihovim očekivanim vrednostima. Poslednje, peto poglavlje sadrži zaključna razmatranja autora ove teze.

Kompletan VHDL kod realizacije, kao i sam tekst rada, će biti priloženi u elektronskoj formi na pratećem CD-u.

2. PROVERA ISPRAVNOSTI IP ZAGLAVLJA I SEGMENTACIJA IP PAKETA

U ovom poglavlju će biti opisana verifikacija ispravnosti IP zaglavlja i segmentacija IP paketa na ćelije. Prvo će, u glavnim crtama, biti objašnjen postupak obrade primljenog paketa, a zatim će detaljno biti prikazana struktura IPv4 paketa, postupak verifikacije ispravnosti zaglavlja kao i segmentacija paketa na ćelije.



Slika 2.1 - Blok šema trećeg sloja OSI referentnog modela prijemnog porta rutera [2].

Slika 2.1 prikazuje pojednostavljenu blok šemu trećeg sloja OSI referentnog modela prijemnog porta rutera. Posle odgovarajuće obrade paketa na drugom sloju, IP paket se pojavljuje na trećem sloju OSI modela.

Prvo se ispituje da li u zaglavlju IP paketa postoje greške. Paralelno sa tom proverom iz zaglavlja se prikupljaju sve informacije neophodne za njegovu dalju obradu. Jedna od tih informacija je i odredišna IP adresa na osnovu koje se vrši lukap (*lookup*) funkcija. Sva dalja obrada IP paketa se vrši samo ako se utvrdi da je zaglavlje paketa ispravno. U suprotnom, paket se odbacuje kao neispravan.

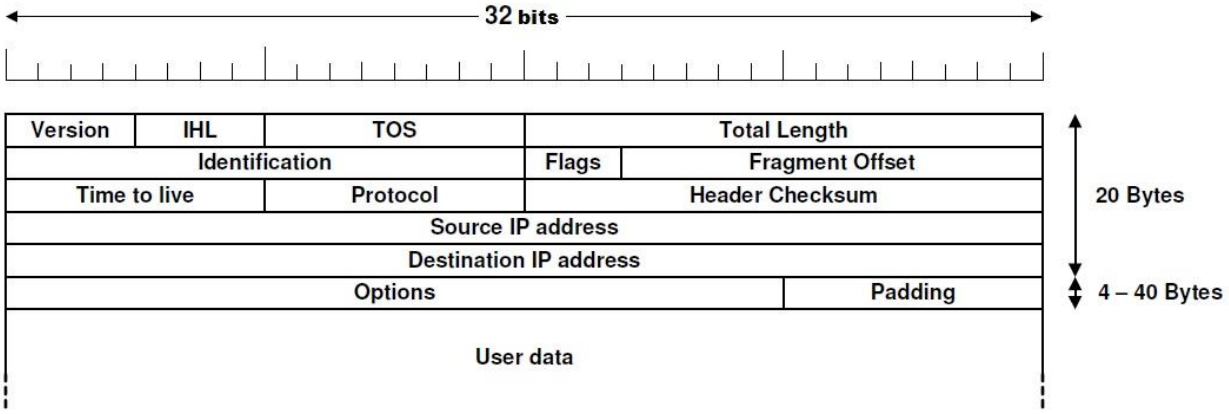
Iz bloka koji vrši proveru IP zaglavlja ispravan paket se šalje u blok koji vrši segmentaciju paketa na ćelije fiksne dužine. Nakon segmentacije, ćelije paketa se prosleđuju ka komutatoru

(sviču). Kontroler komutatora (*scheduler*) na osnovu određenog algoritma za raspoređivanje paketa određuje redosled prosleđivanja ćelija kroz komutator. Kao rezultat lukap funkcije kontroleru se dostavlja i identifikator odredišnog porta na koji bi trebalo proslediti ćelije odgovarajućeg paketa.

2.1. Struktura IP paketa

IP paket se sastoji od dva dela. Prvi deo čini zaglavlje, a drugi deo čine podaci koji se prenose. IP paket na svom kraju ne sadrži polje za verifikaciju ispravnog prenosa kompletnog IP paketa zato što se ta provera već obavlja na nižem sloju prenosa.

Zaglavlje IPv4 paketa se sastoji od 14 polja, od kojih su 13 obavezna. Poslednje polje u zaglavlju je opciono. Polja bitna za ovaj rad će biti objašnjena u nastavku.



Slika 2.2 - Format IPv4 zaglavlja [3].

Polja IPv4 zaglavlja koja su bitna za hardversku implementaciju realizovanu u okviru ove teze su:

- ***Version*** – Verzija IP protokola. Dugačko je četiri bita i za IPv4 ima vrednost 4.
- ***Internet Header Length (IHL)*** – Dužina IP zaglavlja izražena u broju 32-bitnih reči. Pošto zaglavlje može da sadrži različit broj opcionih polja, u ovom polju navedena je dužina zaglavlja. Minimalna vrednost ovog polja je 5, što predstavlja dužinu od 160 bita ili 20 bajtova. Pošto je u pitanju četvorobitno polje, maksimalna dužina zaglavlja je 15 reči, odnosno 480 bita ili 60 bajtova.
- ***Total Length*** – Ukupna dužina čitavog paketa u bajtovima. Minimalna dužina paketa je 20 bajtova (takav paket sadrži samo najkraće zaglavlje dužine 20B) a maksimalna 65,535 bajtova jer je to najveća vrednost koja se može predstaviti pomoću 16 bita.
- ***Time To Live (TTL)*** – Vreme života paketa. Inicijalnu vrednost *TTL* polja postavlja pošiljalac paketa. Preporučena inicijalna vrednost je 64. Prilikom svakog prolaska kroz neki ruter u mreži, vrednost ovog polja se smanjuje za jedan. Ako *TTL* polje dostigne vrednost nula pre nego što paket stigne na odredište, taj paket se odbacuje i ICMP poruka o grešci (11 – *Time Exceeded*) se šalje pošiljaocu. Svrha *TTL* polja je sprečavanje pojave da paketi koji se ne mogu isporučiti beskonačno kruže u mreži.
- ***Header Checksum*** – Polje za proveru ispravnosti IP zaglavlja. Ruter vrši verifikaciju ispravnosti zaglavlja pristiglog paketa uz pomoć kontrolne sume koja se nalazi u polju *Header Checksum*. Ako se prilikom verifikacije kontrolne sume utvrdi da u zaglavlju

postoji greška, taj paket se odbacuje. U slučaju da ruter promeni bilo koje polje zaglavlja, poput dekrementiranja vrednosti *TTL* polja, nova kontrolna suma se izračunava i upisuje u polje *Header Checksum*.

- ***Source IP address*** – Izvorišna IP adresa dužine 32 bita.
- ***Destination IP address*** – Odredišna IP adresa dužine 32 bita.

2.2. Provera ispravnosti zaglavlja IP paketa

Provera ispravnosti IPv4 zaglavlja započinje proverom da li polje *Version* ima vrednost četiri. Zatim se proverava *TTL* polje koje mora imati vrednost veću od nule. Ako *TTL* polje ima vrednost 0 paket se odbacuje. Na kraju se vrši verifikacija kontrolne sume.

Verifikacija kontrolne sume će biti prikazana na sledećem primeru IPv4 zaglavlja (napisanog u heksadecimalnoj notaciji):

45000073000040004011b861c0a80001c0a800c7

Prilikom verifikacije kontrolne sume, zaglavlje se prvo deli na 16-bitne reči. Zatim se te reči sumiraju.

4500 + 0073 + 0000 + 4000 + 4011 + b861 + c0a8 + 0001 + c0a8 + 00c7 = 2fffd

2fffd u binarnoj notaciji:

0010 1111 1111 1111 1101

Pošto rezultat mora biti 16-bitan biti prekoračenja se sabiraju sa najnižih 16 bita:

0010 + 1111 1111 1111 1101 = 1111 1111 1111 1111

Ako se inverzijom rezultata dobije vrednost 0000 0000 0000 0000, smatra se da je zaglavlje bez bitskih grešaka, u suprotnom u zaglavlju postoje bitske greške i paket se odbacuje.

Kao što se vidi u datom primeru nije detektovana greška. Kada bi u zaglavlju postojala bitska greška invertovana vrednost rezultujuće sume bi bila različita od nule.

2.3. Segmentacija IP paketa na ćelije

Kako bi se povećao kapacitet Internet rutera, u njih se instaliraju svičevi (komutatori) velikog kapaciteta koji se koriste u kombinaciji sa naprednim algoritmima za raspoređivanje paketa.

Većina savremenih algoritama za raspoređivanje paketa bazirana je na ćelijama fiksne dužine jer one omogućavaju efikasno iskorišćenje kapaciteta komutatora. Pošto IP paketi nemaju fiksnu dužinu, nakon verifikacije ispravnosti zaglavlja paketa, mora se izvršiti njegova segmentacija na ćelije fiksne dužine. Ako poslednja ćelija ne bude do kraja ispunjena bitima paketa, ona se može dopuniti nulama kako bi imala traženu dužinu.

Svaka ćelija sadrži zaglavlje koje je neophodno za pravilan i efikasan proces rekonstrukcije paketa na izlaznom portu. U zaglavlju se nalazi identifikator izvorišnog porta rutera koji se koristi za razdvajanje ćelija koje pripadaju različitim paketima (jer oni dolaze sa različitih portova). Pored toga, zaglavlje može sadržati identifikator odredišnog porta rutera, i polje koje određuje prioritet paketa. Zaglavlje najčešće sadrži i indikatore prve i poslednje ćelije u paketu, kao i indikator greške u prijemu paketa.

Nakon prolaska kroz komutator, na izlaznom portu rutera se pojavljuju ćelije sa različitih ulaznih portova rutera. Za svaku ćeliju se na osnovu njenog zaglavlja utvrđuje kom IP paketu ona

pripada. Pored toga, na osnovu indikatora poslednje ćelije paketa dobija se informacija kada su sve ćelije jednog paketa stigle na izlazni port.

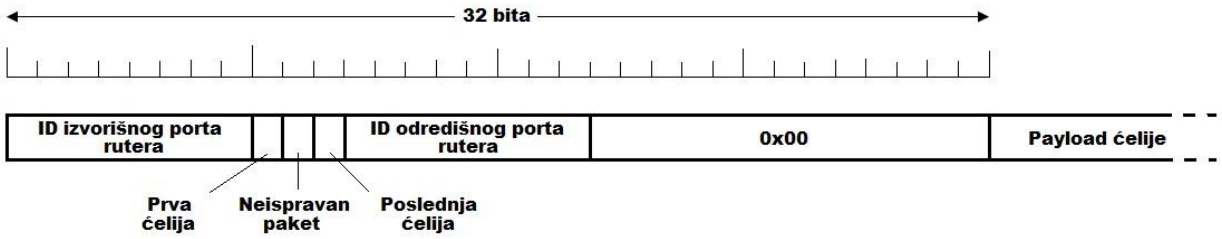
Kad su prikupljene sve ćelije jednog paketa vrši se njegova rekonstrukcija. Zahvaljujući podatku o ukupnoj dužini paketa, koji se nalazi u IP zaglavlju u prvoj ćeliji, nule kojima je dopunjena poslednja ćelija se ne uzimaju u obzir prilikom sastavljanja paketa.

3. OPIS IMPLEMENTACIJE

Za realizaciju implementacije korišćen je VHDL jezik. Ovo poglavlje sadrži detaljan opis strukture realizovanog dizajna i VHDL koda napisanog za potrebe njegove hardverske implementacije.

Dizajn treba da vrši ispitivanje validnosti IPv4 zaglavlja, ekstrakciju odredišne IP adrese i segmentaciju ispravnih IP paketa na ćelije. U jednom ciklusu takta na ulaz dizajna dolazi 64 bita (8 bajtova) paketa pošto se u slučaju 10G portova prijem paketa sa sloja linka podataka vrši preko 64-bitne magistrale podataka. U ovom radu podrazumevamo generalan slučaj - početak i kraj paketa se mogu pojaviti na proizvoljnom bajtu ulaza. Dizajn treba da omogući kontinualan prijem paketa, tj. nakon kraja jednog paketa, na preostalim bajtovima 64-bitne magistrale podataka u istom ciklusu takta, može početi sledeći paket. Na izlazima dizajna treba da se pojavljuju odredišne IP adrese i formirane ćelije ispravnih paketa.

Paketi se dele na ćelije dužine 256 bita. Svaka ćelija treba da sadrži zaglavlje dužine 32 bita. Zaglavlje ćelije treba da sadrži identifikator porta rutera na kom je paket primljen, kao i identifikator odredišnog porta rutera na koji bi trebalo proslediti ćelije odgovarajućeg paketa. Pored ta dva polja, u zaglavlju treba da postoje i tri jednobitna indikatora, za prvu ćeliju paketa, poslednju ćeliju paketa i neispravan paket. Ostali biti zaglavlja treba da imaju vrednost nula. Raspored polja u zaglavlju je prikazan na slici 3.1.



Slika 3.1 - Raspored polja u zaglavlju ćelije.

U nastavku ovog poglavlja prvo će biti opisani interfejsi dizajna, a zatim će detaljno biti opisana struktura dizajna i njegovih pojedinačnih komponenti.

3.1. Interfejsi

Dizajn sadrži ulazne i izlazne interfejse. Ulazni interfejsi su *Reset*, *Clk*, *Paket_in*, *Start* i *Izvorisni_port_ID*, a izlazni su *Odredisna_adresa*, *Preuzmi_adresu*, *Greska_duzina*, *Celija* i *Upisi*.

Signal *Reset* vrši sinhroni reset čitavog dizajna i vraća ga u početno stanje.

Signal *Clk* se koristi kao signal takta. Signal takta za 10G port ima frekvenciju 156,25 MHz.

Signal *Paket_in* predstavlja ulaz širine 64 bita tj. 8 bajtova preko koga se primaju bajtovi paketa.

Signal *Start* označava početak novog IP paketa na ulazu *Paket_in*. Ovaj signal je širine 8 bita, po jedan bit za svaki bajt ulaza *Paket_in*. Početak paketa se može javiti na proizvoljnom bajtu ulaza *Paket_in*. Bit signala *Start* koji odgovara prvom bajtu novog paketa na ulazu *Paket_in* ima vrednost 1 i ta vrednost traje samo jedan takt, tj. pojavljuje se uporedo sa prvim bajtom i vraća na 0 u sledećem taktu.

Signal *Izvorisni_port_ID* je ulaz širine 8 bita i predstavlja identifikator porta rutera koji je primio dati paket.

Signal *Odredisna_adresa*, širine 32 bita, predstavlja odredišnu IPv4 adresu primljenog paketa.

Signal *Preuzmi_adresu* signalizira da je odredišna adresa paketa postavljena na izlaz *Odredisna_adresa*. Ovaj jednobitni signal dobija aktivnu vrednost uporedo sa postavljanjem adrese na izlaz *Odredisna_adresa* i ta vrednost traje samo jedan takt.

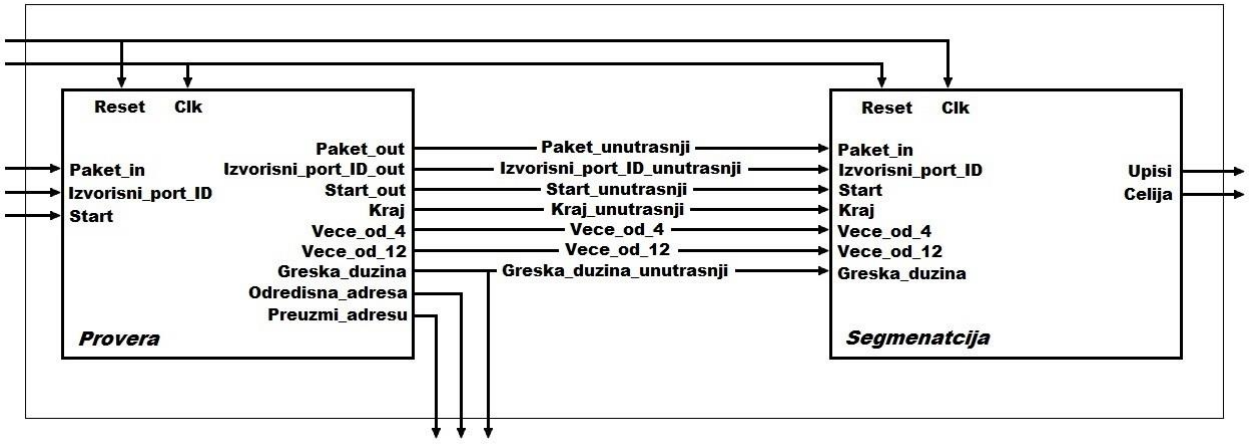
Signal *Greska_duzina* je jednobitni signal koji signalizira grešku u dužini IP paketa. Greška u dužini paketa se javlja ako se početak novog paketa pojavi pre nego što su primljeni svi bajtovi prethodnog paketa, u skladu sa njegovom dužinom definisanom u odgovarajućem polju zaglavlja (*Total Length* polje).

Signal *Celija*, širine 256 bita, predstavlja jednu formiranu ćeliju.

Signal *Upisi* je jednobitni signal i signalizira da je formirana ćelija postavljena na izlaz *Celija*. Dobija aktivnu vrednost uporedo sa postavljanjem ćelije na izlaz *Celija* i ta vrednost traje samo jedan takt.

3.2. Struktura implementacije

Dizajn se sastoji iz dva dela, komponente koja vrši proveru IP zaglavlja i komponente koja vrši segmentaciju paketa na ćelije.



Slika 3.2 - Blok šema dizajna.

Komponenta Provera preuzima bajtove paketa sa ulaza dizajna. Kada svi bajtovi zaglavlja paketa budu primljeni vrši proveru njegove ispravnosti. Ako je zaglavlje ispravno postavlja odredišnu IPv4 adresu paketa na izlaz *Odredisna_adresa* i to signalizira na izlazu

Preuzmi_adresu. Bajtove paketa, vrednost ulaznih signala *Start* i *Izvorisni_port_ID* prosleđuje ka komponenti Segmentacija. Kada budu primljeni svi bajtovi paketa, određuje na kom bajtu (od 8 bajtova, koliko ih u svakom ciklusu takta dolazi na ulaz dizajna) se nalazi kraj paketa i formira signal *Kraj* koji ima iste karakteristike kao signal *Start* (širine 8 bita, bit koji odgovara poslednjem bajtu dobija vrednost 1). Pored toga, formira i signale *Vece_od_4* i *Vece_od_12* koji su kao i signal *Kraj* bitni za proces segmentacije. U slučaju da dođe do greške u dužini paketa, na izlaz *Greska_duzina* se postavlja aktivna vrednost. Pored izlaza dizajna taj signal se prosleđuje i komponenti Segmentacija. Važno je napomenuti da se vrednosti signala *Start*, *Kraj*, *Vece_od_4* i *Vece_od_12* prosleđuju komponenti Segmentacija samo za pakete koji imaju ispravno zaglavlje.

Komponenta Segmentacija preuzima bajtove paketa kao i signale *Start*, *Izvorisni_port_ID*, *Kraj*, *Vece_od_4*, *Vece_od_12* od komponente Provera. Na osnovu tih signala od ispravnih paketa se formiraju ćelije dužine 256 bita. Svaka ćelija ima zaglavlje dužine 32 bita, a ostalih 224 bita se popunjavaju bitima paketa. Formirana ćelija se prosleđuje na izlaz dizajna *Celija* i to se signalizira aktivnom vrednošću na izlazu *Upisi*.

3.2.1. Komponenta Provera

Uzimajući u obzir da se odjednom obrađuju po 64 bita paketa, jer je tolika širina ulazne magistrale, i da dizajn treba da radi na taktu frekvencije 156,25 MHz, provera kontrolne sume je morala biti podeljena na nekoliko koraka sa odgovarajućim međusumama.

Bajtovi sa ulaza dizajna se prepisuju u registar *Pomocni_registar*, i dalje se kroz još sedam registara prenose do izlaza ovog entiteta. Redosled bajtova se tokom ovog prenosa ne menja.

```
Pomocni_registar <= Paket_in;
Registar(0) <= Pomocni_registar;
Registar(1) <= Registar(0);
Registar(2) <= Registar(1);
Registar(3) <= Registar(2);
Registar(4) <= Registar(3);
Registar(5) <= Registar(4);
Registar(6) <= Registar(5);
Paket_out <= Registar(6);
```

Kada na ulaz dizajna stigne početak novog paketa, vrednost signala *Start* se prepisuje u registar *Offset*. Prvi bajtovi paketa se sa ulaza prepisuju u *Pomocni_registar*. U sledećem taktu se na osnovu sačuvane vrednosti signala *Start* u *Offset* registru, iz registra *Pomocni_registar* u registar *Registar_zaglavlje* prepisuju samo prvi bajtovi paketa. Ostatak bajtova registra *Registar_zaglavlje* se popunjava bajtovima sa ulaza, na takav način da se ne naruši redosled bajtova u paketu. Time se postiže da u tom ciklusu takta *Registar_zaglavlje* sadrži prvih osam bajtova paketa. U sledećem taktu se ponovo, na osnovu vrednosti registra *Offset*, iz registra *Pomocni_registar*, u kome se sada nalaze bajtovi koji su u prošlom taktu bili na ulazu, u *Registar_zaglavlje* prepisuju bajtovi koji nisu bili prepisani u prošlom taktu, a ostatak bajtova se ponovo prepisuje sa ulaza. U registru *Registar_zaglavlje* se sada nalaze drugih osam bajtova paketa. Ovaj proces se nastavlja i obezbeđuje da se u registru *Registar_zaglavlje* uvek nalaze uzastopne osmorke bajtova paketa. Provera zaglavlja se zasniva upravo na vrednostima registra *Registar_zaglavlje*.

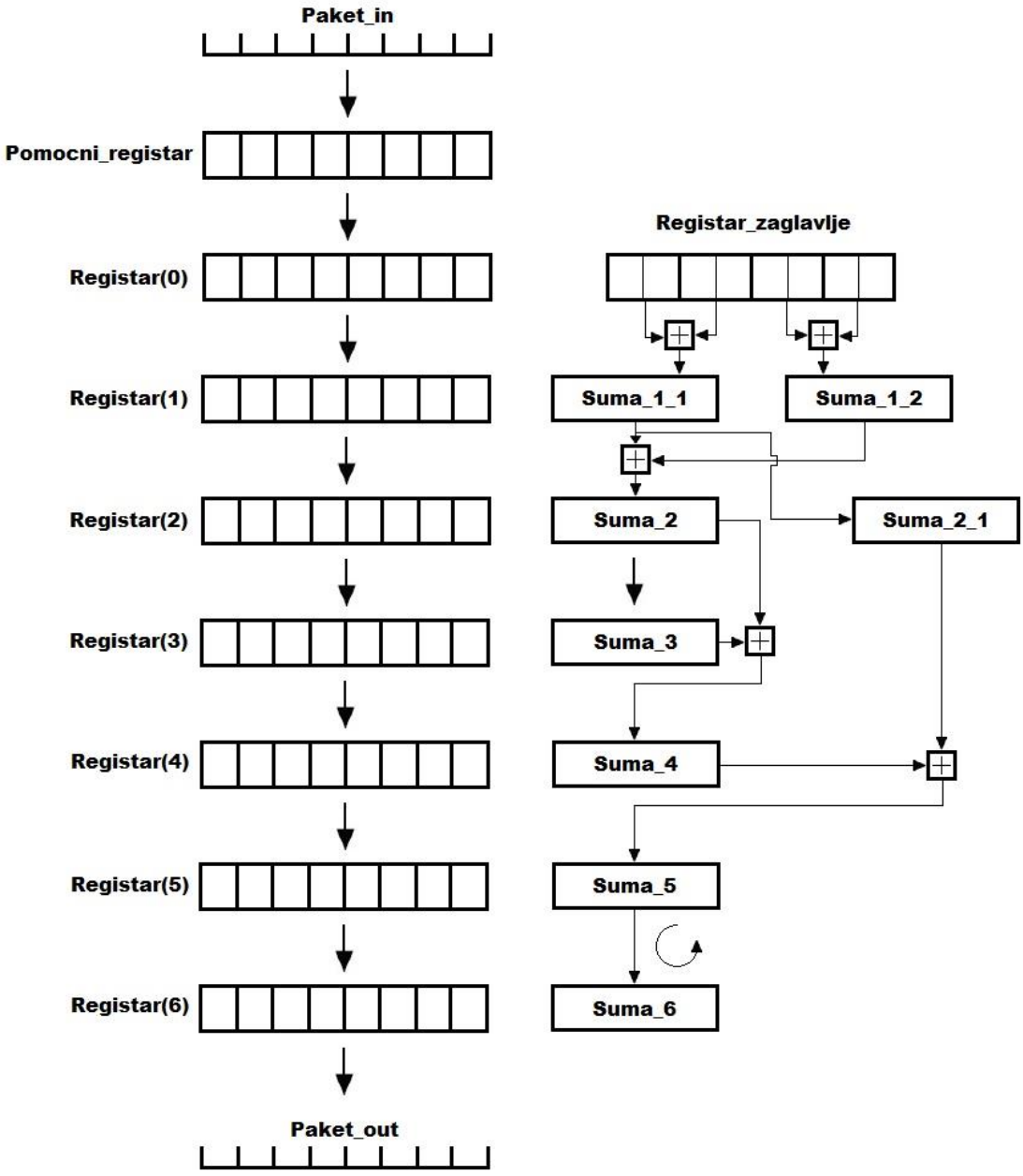
Prilikom formiranja vrednosti registra *Registar_zaglavlje* u kodu se koristi promenljiva *data_in*, dužine 64 bita.

```
CASE (Offset) IS
      WHEN "10000000" => data_in(63 DOWNTO 0) := Pomocni_registar(63 DOWNTO 0);
      WHEN "01000000" => data_in(63 DOWNTO 8) := Pomocni_registar(55 DOWNTO 0);
                         data_in(7 DOWNTO 0) := Paket_in(63 DOWNTO 56);
      WHEN "00100000" => data_in(63 DOWNTO 16) := Pomocni_registar(47 DOWNTO 0);
                         data_in(15 DOWNTO 0) := Paket_in(63 DOWNTO 48);
      WHEN "00010000" => data_in(63 DOWNTO 24) := Pomocni_registar(39 DOWNTO 0);
                         data_in(23 DOWNTO 0) := Paket_in(63 DOWNTO 40);
      WHEN "00001000" => data_in(63 DOWNTO 32) := Pomocni_registar(31 DOWNTO 0);
                         data_in(31 DOWNTO 0) := Paket_in(63 DOWNTO 32);
      WHEN "00000100" => data_in(63 DOWNTO 40) := Pomocni_registar(23 DOWNTO 0);
                         data_in(39 DOWNTO 0) := Paket_in(63 DOWNTO 24);
      WHEN "00000010" => data_in(63 DOWNTO 48) := Pomocni_registar(15 DOWNTO 0);
                         data_in(47 DOWNTO 0) := Paket_in(63 DOWNTO 16);
      WHEN "00000001" => data_in(63 DOWNTO 56) := Pomocni_registar(7 DOWNTO 0);
                         data_in(55 DOWNTO 0) := Paket_in(63 DOWNTO 8);
      WHEN OTHERS => NULL;
END CASE;

Registar_zaglavlje <= data_in;
```

Algoritam računanja kontrolne sume je prikazan na slici 3.3. Dok se prvi bajtovi paketa pojave u registru *Registar(6)* u registru *Suma_6* će se nalaziti rezultat provere kontrolne sume.

Tok sumiranja je sledeći: U trenutku kada prvi bajtovi paketa (u redosledu u kom su se nalazili na ulazu) prelaze iz registra *Pomocni_registar* u registar 0, u registar *Registar_zaglavlje* se upisuje prvih osam bajtova zaglavlja i sumiranje počinje. Kada se prvi bajtovi paketa budu nalazili u registru 2 biće izračunata suma prvih 8 bajtova zaglavlja. U sledećem taktu, prvi bajtovi prelaze u registar 3, i se izračunava se suma drugih 8 bajtova. U trenutku kada prvi bajtovi budu prebačeni u registar 4 biće izračunata suma prvih 16 bajtova zaglavlja. U sledećem taktu prvi bajtovi se prebacuju u registar 5, a prethodno izračunatoj sumi se dodaje i suma poslednjih 4 bajta zaglavlja. Zbog vrednosti bita koji se prenose pri sabiranju, sve međusume u ovom postupku su dužine 20 bita, iako je kontrolna suma dužine 16 bita. Preostaje da se u sledećem taktu izvrši sabiranje prva četiri bita sume (koji sadrže prenesene vrednosti) sa njenim ostalim bitima. Ako se u registru *Suma_6* na poslednjih 16 mesta pojave sve jedinice, to pokazuje da u zaglavlju ne postoje greške.



Slika 3.3 - Algoritam računanja kontrolne sume.

Brojac_zaglavlje broji koliko je 32-bitnih reči zaglavlja (dužina zaglavlja u polju *IHL* je predstavljena brojem 32-bitnih reči) prošlo kroz *Registar_zaglavlje*. Na osnovu tog brojača se iz zaglavlja preuzimaju vrednosti polja *Total_length* i *Destination_address*, i proveravaju vrednosti polja *Version* i *TTL*.

Kako najkraći IP paket ima dužinu od samo 20 bajtova, a paketi mogu u kontinuitetu (čim se završi jedan paket, na sledećem bajtu može počinjati sledeći paket) dolaziti na ulaz, može se pojaviti problem sa računanjem kontrolne sume ako posle kraja kratkog paketa (koji sadrži zaglavlje i eventualno još par bajtova) u istom taktu na preostalim bajtovima ulaza počne sledeći paket. Pošto se nova vrednost signala *Start* upisuje u registar *Offset*, i *Registar_zaglavlje* počinje da vrši „poravnavanje“ za prvih osam bajtova sledećeg paketa, suma poslednjih bajtova

zaglavlja prethodnog, kratkog, paketa neće biti izračunata i dodata prethodnim međusumama u proveri.

Da bi se takva situacija izbegla, umesto jednog koriste se dva bloka za proveru kontrolne sume. Svaki blok sadrži svoje registre *Offset* i *Registar_zaglavlje*. Kao što se u kodu može videti, svi registri jednog bloka imaju u svom nazivu *_I*, a registri drugog bloka *_II*.

Ovi blokovi se koriste naizmenično. Kada se na ulazu pojavi početak novog paketa, na osnovu vrednosti registra *Id_provere* se bira blok koji nije korišćen za prethodni paket. Time se omogućava da jedan blok nastavi da radi sa starom *Offset* vrednošću i pravilno do kraja izvrši sumiranje svih bajtova zaglavlja, dok drugi blok počinje da sumira bajtove novog paketa. Pošto sada postoje dva registra u kojima se može nalaziti rezultat provere kontrolne sume, *Suma_I_6* za prvi blok i *Suma_II_6* za drugi blok, vrednost identifikatora Id_provere se prenosi kroz registre paralelno sa bajtovima paketa.

Zajedno sa bajtovima paketa od ulaza do izlaza ovog entiteta se moraju preneti i odgovarajuće vrednosti signala *Start* i *Izvorisni_port_ID*. Oni se prenose na isti način kao bajtovi paketa, kroz nizove zasebnih registara. Za prenos svih signala se koristi isti broj registara i prenos se odvija sinhronizovano kako bi se kombinacije signala sa ulaza neizmenjene pojavljivale i na izlazu.

Pored vrednosti signala *Start* i *Izvorisni_port_ID*, na isti način, paralelno sa bajtovima paketa se prenose i vrednost identifikatora *Id_provere*, odredišna IPv4 adresa, vrednosti signala *Kraj*, *greska* i *greska_duzina*.

Kada prvi bajtovi paketa stignu u *Registar(6)*, njihova odgovarajuća vrednost signala *Start* će stići u *Registar_start(6)*. U tom trenutku se na osnovu vrednosti u *Registar_Id_provere* ispituje sadržaj odgovarajućeg registra koji sadrži rezultat provere kontrolne sume. Pored toga proverava se i *Registar_greska(6)*, koji će imati aktivnu vrednost u slučaju neispravne vrednosti *Version* ili *TTL* polja.

Ako je kontrolna suma ispravna i *Registar_greska(6)* nema aktivnu vrednost, zaglavlje paketa je ispravno i vrednost *Registar_start(6)* se prosleđuje na izlaz *Start_out*, zajedno sa prvim bajtovima paketa koji se pojavljuju na izlazu *Paket_out*. Odredišna IPv4 adresa paketa se šalje na izlaz *Odredisna_adresa*, a izlaz *Preuzmi_adresu* se postavlja na aktivnu vrednost.

Ako zaglavlje paketa nije ispravno, bajtovi paketa se i dalje prosleđuju na izlaz *Paket_out*, ali se na izlaze *Start_out*, *Kraj* i *Odredisna_adresa* ne šalju odgovarajući signali, pa se za takve pakete ne vrši nikakva dalja obrada.

Glavni brojač u ovom entitetu (signal *Brojac*) broji koliko bajtova trenutnog paketa je primljeno. Na početku paketa on se postavlja na vrednost broja primljenih bajtova paketa u prvom taktu (u kom je započet prijem paketa), a u sledećim taktovima se uvećava za osam.

Kraj paketa se može registrovati na dva načina. Prvi način je kada *Brojac* dobije vrednost veću od vrednosti polja *Total_length* u zaglavlju paketa. Tada se računa razlika brojača i dužine paketa i na osnovu nje se određuje pozicija poslednjeg bajta. Na osnovu pozicije poslednjeg bajta formira se signal *Kraj*. Drugi način predstavlja pojavljivanje početka sledećeg paketa, koji se signalizira odgovarajućom vrednošću signala *Start*, u istom taktu u kom se nalazi i kraj trenutnog paketa. Tada se određuje koliko bajtova u tom taktu ne pripada novom paketu. Ako je zbir tog broja i broja prethodno primljenih bajtova veći od ukupne dužine paketa, tada se pozicija poslednjeg bajta i vrednost signala *Kraj* određuju na osnovu razlike kao u prethodnom slučaju. Ako je ukupan broj primljenih bajtova manji od dužine paketa, aktivira se signal za grešku u dužini.

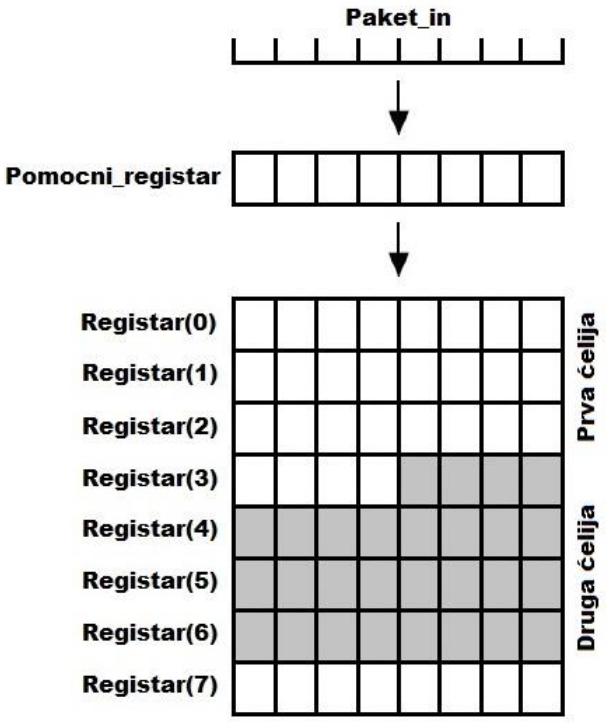
Vrednost signala *Vece_od_4* i *Vece_od_12* se formira na osnovu vrednosti signala *Offset* i *Kraj* za dati paket. Za signal *Offset* određuje se broj bita sa desne strane bita koji ima vrednost 1, uključujući i njega. I za signal *Kraj* se određuje broj bita, ali sa leve strane bita koji ima vrednost 1, uključujući i njega. Zatim se određuje ukupan broj bita kao zbir ta dva broja. Taj zbir predstavlja broj poslednjih bajtova paketa. Ako je zbir veći od 4 signal *Vece_od_4* dobija aktivnu vrednost, a ako je veći od 12, i signal *Vece_od_12* će biti aktivan. Ovi signali se pojavljuju na izlazu zajedno sa signalom *Kraj* i njihova svrha će biti objašnjena u delu teksta u kome se opisuje segmenatcija.

3.2.2. Komponenta Segmentacija

Komponenta Segmentacija od bajtova paketa sa ulaza formira ćelije dužine 256 bita. Svaka ćelija ima zaglavlje dužine 32 bita i *payload* deo dužine 224 bita.

Najbitniji delovi ovog entiteta su registar *Pomocni_registar* i osam registara u koje se smeštaju ćelije paketa. Ovi registri su dužine 8 bajtova, jer je tolika širina ulaza *Paket_in*. Za smeštanje *payload*-a jedne ćelije, koji ima dužinu od 28 bajtova, potrebna su tri i po registra.

Bajtovi paketa se sa ulaza prepisuju u *Pomocni_registar* (slika 3.4). *Pomocni_registar* ima sličnu ulogu kao u komponenti Provera.



Slika 3.4 – Raspored ćelija u registrima.

Kada se pojavi početak novog paketa, njegovi prvi bajtovi se sa ulaza prepisuju u *Pomocni_registar*. Vrednost ulaza *Start* za dati paket se čuva u registru *Offset*. U sledećem taktu, kada je pristiglo više od osam bajtova paketa koliko je potrebno za upis u registre ćelija, iz registra *Pomocni_registar* se na osnovu vrednosti *Offset* registra uzimaju samo prvi bajtovi paketa. Ostali bajtovi koji čine prvih osam bajtova paketa se uzimaju sa ulaza *Paket_in*. Nakon

njihovog upisivanja u neki od registara, u sledećem taktu se iz registra *Pomocni_registar*, u kome se u tom trenutku nalaze bajtovi koji su u prošlom taktu bili na ulazu *Paket_in*, uzimaju bajtovi koji nisu bili upisani u prošlom taktu, opet na osnovu *Offset* vrednosti. Ostali bajtovi koji čine drugih osam bajtova paketa se ponovo uzimaju sa ulaza. Ovaj proces se nastavlja i na isti način se i ostali bajtovi paketa upisuju u ćelije.

Payload jedne ćelije ima dužinu (u bajtovima) koja nije celobrojni umnožak broja osam, što je broj bajtova koji paralelno dolaze na ulaz, a toliku dužinu imaju i registri u koje se smeštaju ćelije, što dovodi do toga da ćelija zauzima samo polovinu poslednjeg registra.

Kako bi se prilikom segmentacije jednog paketa na što efikasniji način vršio prelazak sa upisa u kompletiranu ćeliju na upis u novu, sledeću ćeliju, dve ćelije su u registrima smeštene jedna za drugom i dele jedan zajednički registar. Tako se, prilikom upisivanja bajtova jednog paketa, upis u granični registar ne razlikuje od upisa u bilo koji drugi registar i ne mora se dodatno voditi računa o granici ćelije.

Registar u koji će se vršiti upis određen je signalom *Brojac*, koji može imati vrednosti od 0 do 7. Na početku, bajtovi paketa se upisuju u registre prve ćelije, počevši od registra sa brojem 0 pa do registra sa brojem 3. Prilikom upisa u registar 3 kompletira se prva ćelija i započinje upis u drugu. Upis u drugu ćeliju se nastavlja upisom u registar 4, a za to vreme se prvoj ćeliji dodaje zaglavlje i ona se šalje na izlaz. Upisom u registar sa brojem 6 se kompletira *payload* druge ćelije, a *Brojac* se ponovo vraća na vrednost 0. Bajtovi paketa se zatim ponovo upisuju u registre prve ćelije dok se druga ćelija šalje na izlaz. Registar sa brojem 7 se koristi samo u posebnim slučajevima prilikom prelaska sa segmentacije jednog paketa na segmentaciju sledećeg, i o tome će biti više reči u daljem tekstu.

Funkcija upisa bajtova paketa u registre realizovana je kodom koji je prikazan ispod. U njemu se, slično kao i u kodu komponente Provera, koristi 64-bitna promenljiva *data_in*.

```
Pomocni_registar <= Paket_in;


CASE (Offset) IS
      WHEN "10000000" => data_in(63 DOWNTO 0) := Pomocni_registar(63 DOWNTO 0);
      WHEN "01000000" => data_in(63 DOWNTO 8) := Pomocni_registar(55 DOWNTO 0);
                         data_in(7 DOWNTO 0) := Paket_in(63 DOWNTO 56);
      WHEN "00100000" => data_in(63 DOWNTO 16) := Pomocni_registar(47 DOWNTO 0);
                         data_in(15 DOWNTO 0) := Paket_in(63 DOWNTO 48);
      WHEN "00010000" => data_in(63 DOWNTO 24) := Pomocni_registar(39 DOWNTO 0);
                         data_in(23 DOWNTO 0) := Paket_in(63 DOWNTO 40);
      WHEN "00001000" => data_in(63 DOWNTO 32) := Pomocni_registar(31 DOWNTO 0);
                         data_in(31 DOWNTO 0) := Paket_in(63 DOWNTO 32);
      WHEN "00000100" => data_in(63 DOWNTO 40) := Pomocni_registar(23 DOWNTO 0);
                         data_in(39 DOWNTO 0) := Paket_in(63 DOWNTO 24);
      WHEN "00000010" => data_in(63 DOWNTO 48) := Pomocni_registar(15 DOWNTO 0);
                         data_in(47 DOWNTO 0) := Paket_in(63 DOWNTO 16);
      WHEN "00000001" => data_in(63 DOWNTO 56) := Pomocni_registar(7 DOWNTO 0);
                         data_in(55 DOWNTO 0) := Paket_in(63 DOWNTO 8);
      WHEN OTHERS => NULL;
END CASE;
```

```
Brojac <= Brojac + "001";

CASE (Brojac) IS
      WHEN "000" =>     Registar(0) <= data_in;
      WHEN "001" =>     Registar(1) <= data_in;
                        IF (Osmi_registar = '1') THEN
                              Registar(0) <= Registar(7);
                        END IF;
      WHEN "010" =>     Registar(2) <= data_in;
      WHEN "011" =>     Registar(3) <= data_in;
                        Kompletna_celija <= '1';
                        Indikator_celije <= "01";
      WHEN "100" =>     Registar(4) <= data_in;
      WHEN "101" =>     Registar(5) <= data_in;
      WHEN "110" =>     Registar(6) <= data_in;
                        Kompletna_celija <= '1';
                        Indikator_celije <= "10";
                        Brojac <= "000";
      WHEN "111" =>     Registar(7) <= data_in;
                        Osmi_registar <= '1';
                        Brojac <= "001";
      WHEN OTHERS => NULL;
END CASE;
```

Kao što se može primetiti iz datog koda, kada je neka ćelija popunjena do kraja i treba da se postavi na izlaz to se signalizira aktivnom vrednošću signala *Kompletna_celija*. Pored ovog signala neophodan je i signal *Indikator_celije* kako bi se znalo koja od dve ćelije je kompletirana.

Signal *Indikator_celije* može imati jednu od tri vrednosti: „01“ ako na izlazu treba da se ispiše prva ćelija, „10“ ako treba da se ispiše druga ćelija ili „11“ ako obe ćelije treba jedna za drugom da se postave na izlaz u sukcesivnim taktovima. Poslednja vrednost se koristi kada se poslednji bajtovi paketa upisuju baš u registar sa brojem 3, koji predstavlja granicu između dve ćelije. U slučaju da više od četiri bajta treba da se upišu, biće neophodne obe ćelije da bi se preneli svi bajtovi paketa.

Važno je napomenuti i da segmentacija paketa uvek počinje od prve ćelije i registra sa brojem 0. Kada se pojavi vrednost ulaza *Start* različita od nule ona se, kao što je već pomenuto, prepisuje u registar *Offset*. Vrednost ulaza *Izvorisni_port_ID* se čuva u registru *Port_id*, a indikator prve ćelije dobija aktivnu vrednost, koja je potrebna prilikom formiranja zaglavlja prve ćelije.

Vrednost ulaza *Kraj* različita od nule signalizira da se na ulazu *Paket_in* nalaze poslednji bajtovi trenutnog paketa. Kako se prilikom upisivanja bajtova u ćelije, na osnovu vrednosti u registru *Offset*, određeni broj bajtova uzima iz registra *Pomocni_registar* a ostatak sa ulaza *Paket_in*, prvo se porede vrednosti signala *Kraj* i *Offset* kako bi se utvrdilo da li će svi bajtovi paketa sa ulaza biti obuhvaćeni upisom u trenutnom ciklusu takta.

Ako to jeste slučaj, bajtovi se upisuju u odgovarajući registar trenutne ćelije, *Brojac* se resetuje na nulu, *Indikator_celije* dobija vrednost koja odgovara trenutnoj ćeliji, a signal *Kompletna_celija* dobija aktivnu vrednost. Time se završava upis u datu ćeliju. Preostali bajtovi te ćelije (ako ih ima) će sadržati neke nasumične vrednosti koje pripadaju prošlim ćelijama, ali to nije bitno jer se prilikom rekonstrukcije paketa uzima u obzir njegova dužina koja je navedena u IP

zaglavlju, pa bajtovi kojima je poslednja ćelija popunjena do tražene dužine nisu od važnosti. Ako je registar u koji se upisuju poslednji bajtovi paketa registar sa brojem 3 proverava se vrednost signala *Vece_od_4*. Aktivna vrednost ovog signala označava da se upisuje više od četiri bajta i da su neophodne obe ćelije da bi se preneli svi bajtovi paketa, pa *Indikator_celije* dobija vrednost „11“. Pored toga, indikator poslednje ćelije dobija vrednost 1 kako bi prilikom formiranja zaglavlja ćelije bit koji označava poslednju ćeliju dobio aktivnu vrednost.

Ako svi bajtovi paketa sa ulaza *Paket_in* nisu obuhvaćeni upisom u trenutnom taktu, upis preostalih bajtova će se obaviti u sledećem taktu, a to se signalizira aktivnom vrednošću signala *Poslednji_bajtovi*. Ovaj signal je neophodan jer signal *Kraj* u sledećem taktu neće imati aktivnu vrednost. Pošto paralelno sa signalom *Kraj* dolaze i validne vrednosti signala *Vece_od_4* i *Vece_od_12*, u trenutnom taktu se mora proveriti i da li signal *Vece_od_12* ima aktivnu vrednost. Slično kao u prethodnom slučaju kada se proveravalo da li se upisuje više od četiri bajta, u ovom slučaju je od interesa da li je broj poslednjih bajtova paketa veći od 12. U trenutnom taktu će se izvršiti upis osam bajtova, pa se time proverava da li za sledeći, poslednji upis ostaje više od četiri bajta. Upis tih poslednjih bajtova u sledećem taktu će se izvršiti na isti način kao i u prethodnom slučaju: bajtovi će biti upisani u jedan od registara, *Brojac* će biti resetovan, *Indikator_celije* će dobiti odgovarajuću vrednost, signal *Kompletna_celija* će dobiti aktivnu vrednost i indikator poslednje ćelije će dobiti vrednost 1. Ako poslednji bajtovi budu upisani u registar 3 *Indikator_celije* će dobiti vrednost u zavisnosti od vrednosti signala *Vece_od_12* u prethodnom taktu.

```
IF (Kraj /= "00000000") THEN
      IF (Offset < Kraj) THEN
            IF (Start = "00000000") THEN
                  Aktivan_paket <= '0';
            END IF;
            Brojac <= "000";
            Kompletna_celija <= '1';
            Poslednja_celija <= '1';
            IF (Brojac(2) = '0') THEN
                  Indikator_celije <= "01";
            ELSE
                  Indikator_celije <= "10";
            END IF;

            IF (Brojac = "011") THEN
                  IF (Vece_od_4 = '1') THEN
                        Indikator_celije <= "11";
                  END IF;
            END IF;
      ELSE
            Poslednji_bajtovi <= '1';

            IF (Vece_od_12 = '1') THEN
                  Dve_celije <= '1';
            END IF;

      END IF;
      IF (Greska_duzina = '1') THEN
            Greska <= '1';
      END IF;

END IF;
```

```
IF (Poslednji_bajtovi = '1') THEN
      IF (Prvi_bajtovi = '1') THEN
            Brojac <= "001";
      ELSE
            IF (Start = "00000000") THEN
                  Aktivan_paket <= '0';
            END IF;
            Brojac <= "000";
      END IF;
      Kompletna_celija <= '1';
      Poslednja_celija <= '1';
      IF (Brojac(2) = '0') THEN
            Indikator_celije <= "01";
      ELSE
            Indikator_celije <= "10";
      END IF;
      IF (Brojac = "011" AND Dve_celije = '1') THEN
            Indikator_celije <= "11";
      END IF;
      Poslednji_bajtovi <= '0';
END IF;
```

U slučaju da je došlo do greške u dužini paketa, komponenta Provera to signalizira signalom *Greska_duzina* paralelno sa signalom *Kraj*, pa u tom slučaju indikator *Greska* dobija aktivnu vrednost kako bi prilikom formiranja zaglavlja poslednje ćelije bit koji označava neispravan paket dobio vrednost 1.

Ćelija označena vrednošću signala *Indikator_celije* se postavlja na izlaz kada signal *Kompletna_celija* ima vrednost 1. Prilikom postavljanja ćelije na izlaz, formira se njeno zaglavlje. Prvih osam bita zaglavlja ćelije dobijaju vrednosti koje odgovaraju identifikatoru izvorišnog porta za paket čiji segment ćelija sadrži. Sledeća tri bita služe za označavanje prve ćelije paketa, neispravnog paketa i poslednje ćelije paketa, respektivno. Ovi tri bita dobijaju vrednosti na osnovu kontrolnih indikatora *Prva_celija*, *Poslednja_celija* i *Greska*. Sledećih osam bita, koji predstavljaju identifikator odredišnog porta se postavljaju na neaktivne vrednosti, pošto se vrednost ovog polja određuje u okviru IP lukap funkcije i upisuje u zaglavlje ćelije van realizovanog bloka za segmentaciju. Preostalih 13 bita do kraja zaglavlja se postavljaju na neaktivne vrednosti.

U slučaju da se kraj trenutnog i početak sledećeg paketa pojave u istom taktu, vrednost signala *Start* novog paketa se čuva u pomoćnom registru *Novi_offset*. Na osnovu te vrednosti prvih osam bajtova novog paketa se upisuju u registar sa brojem 7, dok se poslednji bajtovi prethodnog paketa na osnovu vrednosti *Offset* registra (koji i dalje ima staru vrednost) upisuju u odgovarajući registar poslednje ćelije. Time se završava segmentacija prethodnog paketa, i poslednja ćelija se šalje na izlaz. U sledećem ciklusu takta, vrednost iz registra *Novi_offset* se prepisuje u registar *Offset*. Upisivanje bajtova novog paketa se nastavlja od registra sa brojem 1, dok se u registar 0 prepisuje sadržaj registra 7.

4. VERIFIKACIJA DIZAJNA I OPIS PERFORMANSI

U ovom poglavlju će biti detaljno opisana verifikacija dizajna. Za potrebe verifikacije korišćena je funkcionalna simulacija u okviru ISim simulatora. Zatim će biti prikazani rezultati procesa analize i sinteze u ISE razvojnom okruženju za FPGA čipove proizvođača Xilinx.

4.1. Verifikacija dizajna

ISE softverski paket za funkcionalnu simulaciju rada dizajna koristi ugrađeni ISim simulator. Da bi se izvršila simulacija rada dizajna neophodno je kreirati odgovarajući testbenč. Testbenč predstavlja simulacioni VHDL entitet, koji okružuje dizajn koji se testira i koji generiše odgovarajuće pobude na ulazima dizajna na osnovu kojih se generiše scenario kojim se želi proveriti ispravnost rada dizajna.

Za potrebe kreiranja testbenča nekoliko IP paketa realnog internet saobraćaja je snimljeno Wireshark softverskim alatom.

Za verifikaciju dizajna odabrano je nekoliko test scenarija u kojima su ispravni i neispravni paketi različitim redosledom i sa različitim međusobnim razmakom dovođeni na ulaz dizajna.

Pored izlaza dizajna (*Odredisna_adresa*, *Preuzmi_adresu*, *Greska_duzina*, *Celija*, *Upisi*), posmatrani su i unutrašnji signali koji povezuju dve komponente kako bi se proverilo funkcionisanje svake komponente pojedinačno u svakom od scenarija.

Na početku su bajtovi jednog, ispravnog paketa postavljani na ulaz dizajna kako bi se proverilo da li će njegova segmenatcija biti izvršena pravilno. Posmatrano je i da li je odredišna IP adresa paketa ispravno izdvojena iz zaglavlja pri proveri.

Zatim su na ulaz dizajna dovedena tri ispravna paketa u kontinuitetu (u istom taktu kraj jednog i početak sledećeg), kako bi se proverilo funkcionisanje dizajna kada između paketa nema razmaka.

U sledeća dva slučaja korišćena su dva veoma kratka paketa (dužine 24 bajta), u kontinuitetu, i sa jednim taktom razmaka između njih, kako bi se proverilo da li dizajn na pravilan način vrši obradu najkraćih paketa.

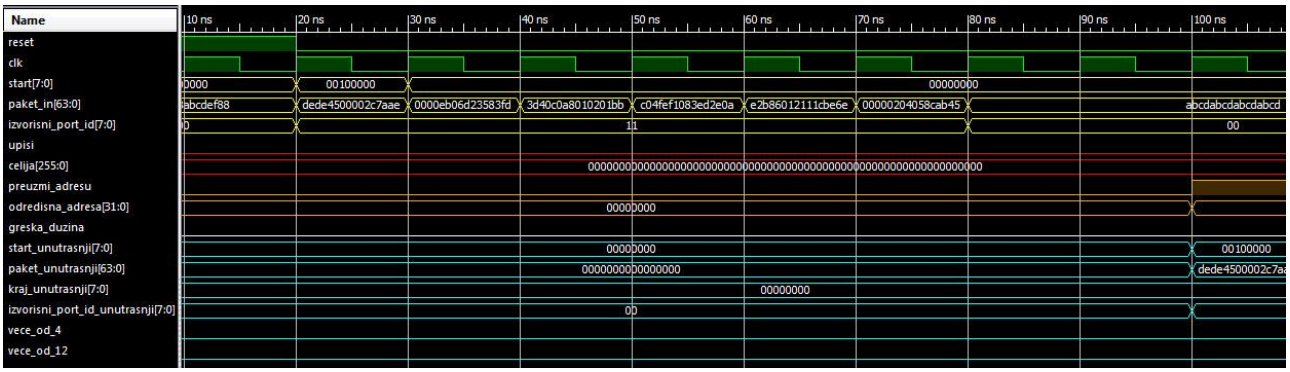
Zatim je ispitano funkcionisanje provere zaglavlja dovođenjem neispravnih paketa na ulaz dizajna. Prvo su na ulaz dovođena po dva paketa, jedan neispravan i za njim jedan ispravan. U prvom slučaju nije bilo razmaka između paketa, dok je u drugom razmak postojao. Zatim su slično kao u jednom od prethodnih slučajeva, na ulaz postavljana tri paketa zaredom, ali u ovom slučaju drugi paket po redu je imao neispravno zaglavlje.

Na kraju je proverena funkcionalnost detektovanja greške u dužini paketa.

Zbog analogije u principu testiranja, u daljem tekstu će biti prikazani rezultati funkcionalne simulacije samo za nekoliko najbitnijih test scenarija.

Na slici 4.1 se može videti da nakon aktiviranja signala *Reset* svi signali dobijaju inicijalne vrednosti. Zatim, na ulaz *Paket_in* počinju da dolaze bajtovi IP paketa koji ima ispravno zaglavlje.

Na svim slikama, ulazi dizajna će biti prikazani žutom bojom. Može se videti da provera IP zaglavlja traje osam taktova, i da se nakon provere bajtovi paketa pojavljuju na unutrašnjoj magistrali *Paket_unutrasnji*. Unutrašnji signali koji povezuju dve komponente dizajna će na svim slikama biti prikazani plavom bojom.

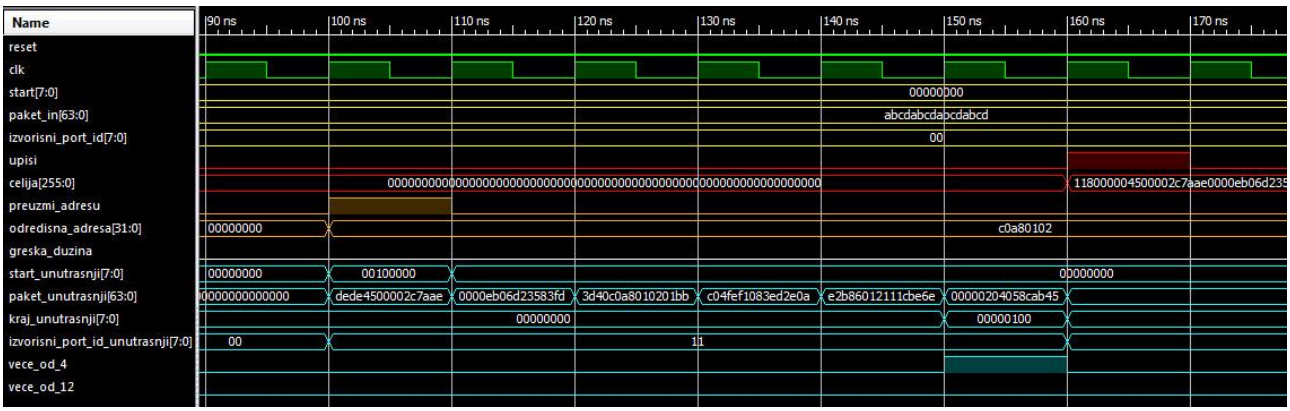


Slika 4.1 – Izgled prozora ISim simulatora, ispravan paket na ulazu - deo I

Na sledećoj slici, slici 4.2, se vidi da je proverom utvrđeno da je zaglavlje paketa ispravno, i da je u istom ciklusu takta u kom su i prvi bajtovi paketa napustili komponentu Provera, njegova odredišna IP adresa postavljena na odgovarajući izlaz, zajedno sa aktivnom vrednošću signala *Preuzmi_adresu* (signali prikazani narandžastom bojom). Pošto je u pitanju ispravan paket, signal *Start_unutrasnji* dobija vrednost koja odgovara datom paketu. Time se započinje proces njegove segmentacije.

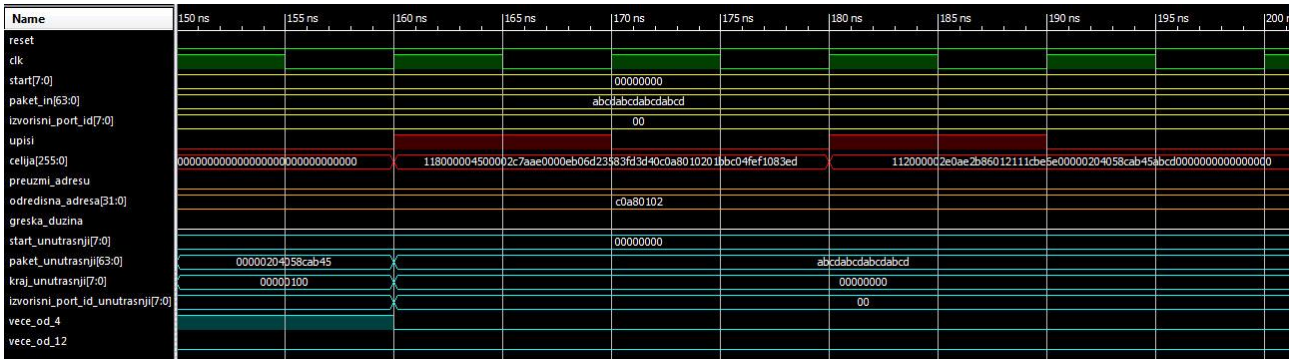
Može se videti da se bajtovi paketa na unutrašnjoj magistrali prenose istim redosledom kojim su se pojavljivali na ulazu dizajna, zajedno sa odgovarajućom vrednošću signala *Izvorisni_port_ID*.

Zajedno sa poslednjim bajtovima paketa komponenta Provera na magistralu *Kraj_unutrašnji* postavlja odgovarajuću vrednost signala *Kraj*, kao i odgovarajuće vrednosti signala *Vece_od_4* i *Vece_od_12*.



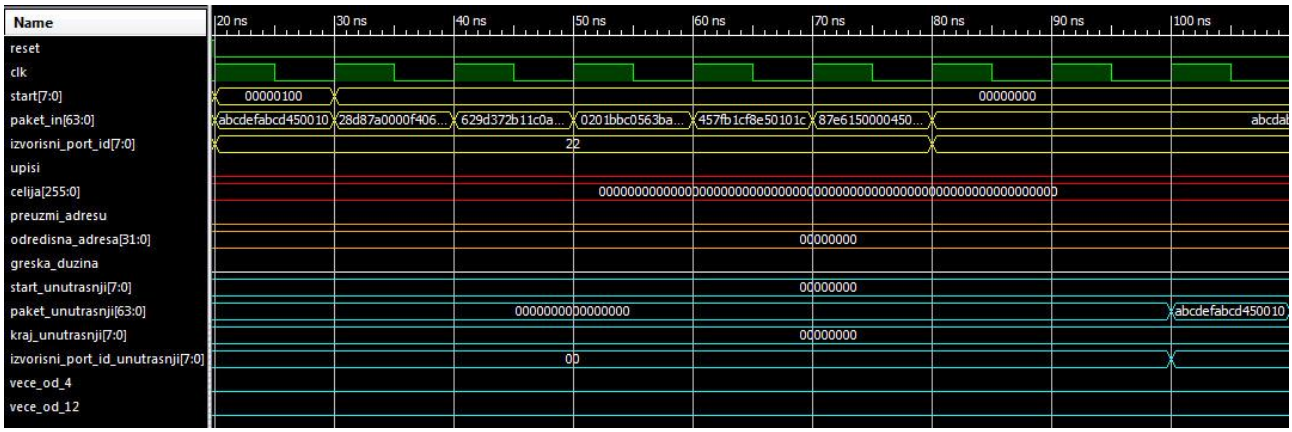
Slika 4.2 - Izgled prozora ISim simulatora, ispravan paket na ulazu - deo II

Kada je ćelija kompletirana ona se pojavljuje na izlazu *Celija*. Postavljanje ćelije na izlaz je signalizirano aktivnom vrednošću signala *Upisi* (signali prikazani crvenom bojom). Na slici 4.3, može se primetiti i da su zaglavlja ćelija ispravno formirana, tj. da prva ćelija sadrži oznaku prve ćelije, poslednja oznaku poslednje, a obe identifikator izvorišnog porta.



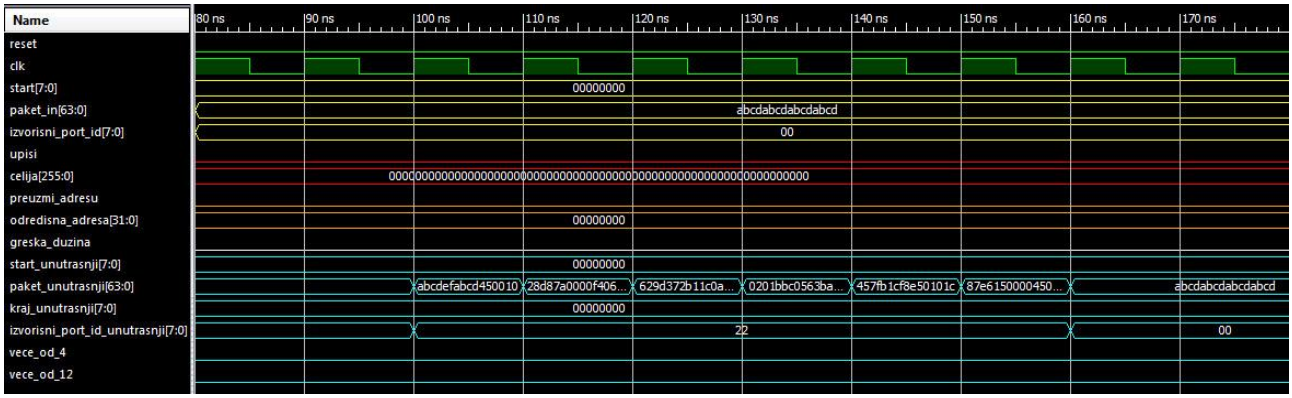
Slika 4.3 - Izgled prozora ISim simulatora, ispravan paket na ulazu - deo III

U sledećem slučaju, slika 4.4, na ulaz dizajna počinju da dolaze bajtovi paketa sa neispravnim zaglavljem. Komponenta Provera bi trebalo da spreči dalju obradu takvog paketa, a rezultati se vide na slici 4.5.



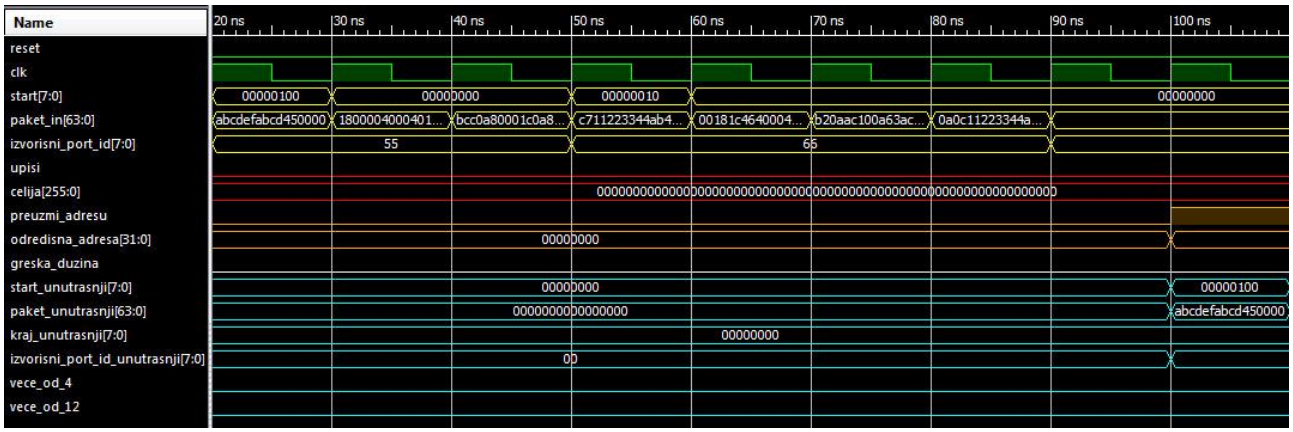
Slika 4.4 - Izgled prozora ISim simulatora, neispravan paket na ulazu - deo I

Nakon izvršene provere zaglavlja bajtovi paketa se pojavljuju na unutrašnjoj magistrali. Pošto u ovom slučaju nije potvrđena ispravnost zaglavlja, njegova odredišna IP adresa nije postavljena na izlaz. Za razliku od prethodnog slučaja, komponenti Segmentacija nisu prosleđene vrednosti signala *Start* i *Kraj* za dati paket, pa zbog toga segmentacija tog paketa nije izvršena.



Slika 4.5 - Izgled prozora ISim simulatora, neispravan paket na ulazu - deo II

Sledeći test scenario, slika 4.6, predstavljaju dva ispravna paketa koji jedan za drugim dolaze na ulaz dizajna. U istom taktu u kom se nalazi kraj jednog, nalazi se i početak sledećeg paketa.

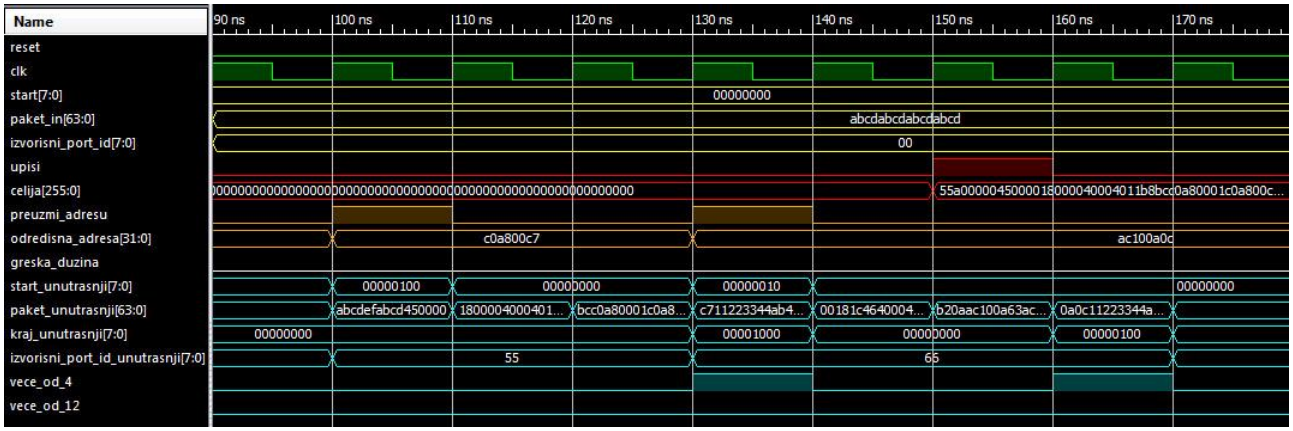


Slika 4.6 - Izgled prozora ISim simulatora kada su na ulazu dva paketa bez razmaka- deo I

Pošto je potvrđena ispravnost zaglavlja prvog paketa, zajedno sa njegovim prvim bajtovima i identifikatorom izvorišnog porta, komponenti Segmentacija je prosleđena i vrednost *Start* signala. Paralelno s tim, njegova odredišna adresa je postavljena na odgovarajući izlaz.

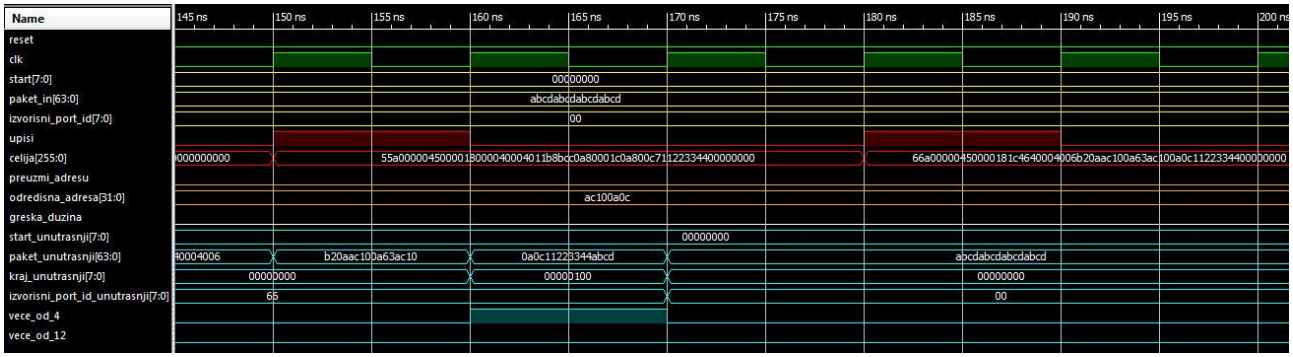
Na granici između dva paketa, u istom taktu se pojavila ispravna vrednost signala *Kraj* za prvi paket, kao i vrednost signala *Start* za sledeći jer je i za njegovo zaglavlje rezultat provere bio pozitivan. Pored toga, i odredišna adresa drugog paketa je postavljena na izlaz.

Zajedno sa poslednjim bajtovima drugog paketa postavljena je i odgovarajuća vrednost signala *Kraj*, a u međuvremenu se na izlazu pojavila i kompletirana ćelija prvog paketa, što se bolje može videti na slici 4.8.



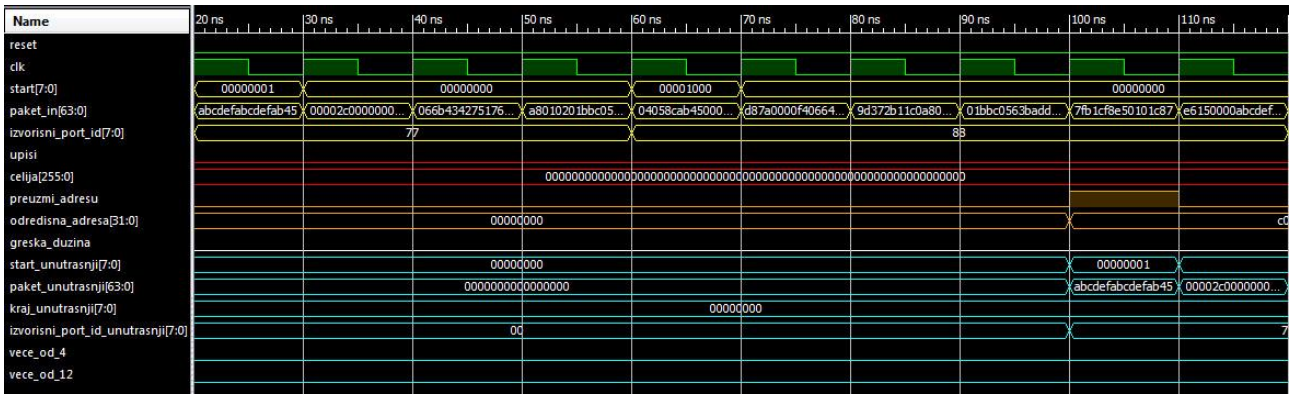
Slika 4.7 - Izgled prozora ISim simulatora kada su na ulazu dva paketa bez razmaka- deo II

Kako su u pitanju kratki paketi, ceo paket može da stane u *payload* jedne ćelije. Na izlaz dizajna je za oba paketa postavljena po jedna ćelija koja u zaglavlju istovremeno sadrži oznake prve i poslednje ćelije.



Slika 4.8 - Izgled prozora ISim simulatora kada su na ulazu dva paketa bez razmaka- deo III

Sledeći test ponovo sadrži dva paketa sa ispravnim zaglavljima. U ovom testu, u zaglavlju prvog paketa navedena je njegova ukupna dužina koja iznosi 44 bajta, dok se početak drugog paketa pojavljuje nakon 29 bajtova od početka prvog paketa. Greška u dužini bi trebalo da bude detektovana i signalizirana na izlazu *Greska_dužina*, a zaglavlje poslednje ćelije prvog paketa bi trebalo da sadrži oznaku za neispravan paket.

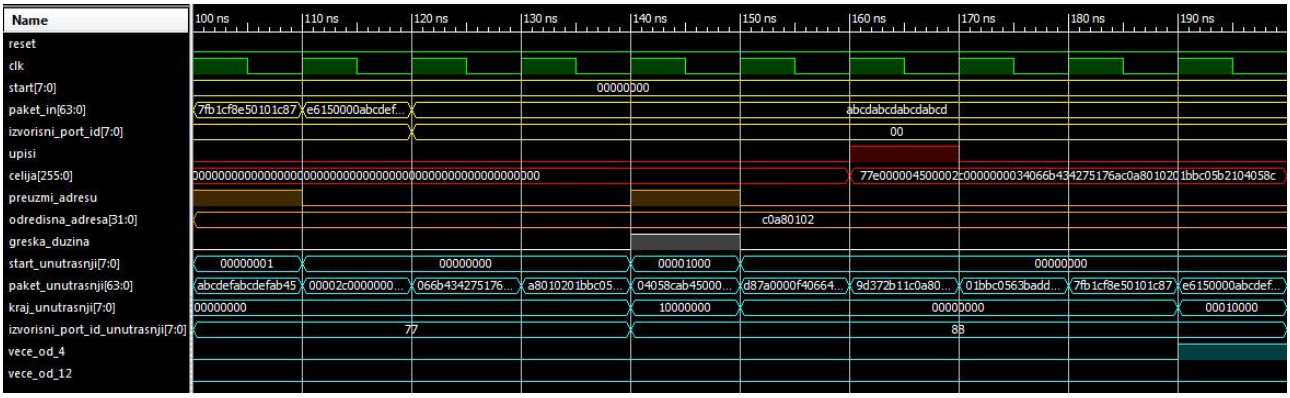


Slika 4.9 - Izgled prozora ISim simulatora, detekcija greške u dužini paketa- deo I

Na slici 4.10 se može videti da je potvrđena ispravnost zaglavlja prvog paketa i da je zajedno sa njegovim prvim bajtovima prosleđena i odgovarajuća vrednost signala *Start*, dok je odredišna adresa postavljena na izlaz.

U trenutku kada se prosleđuje signal *Start* sledećeg paketa, kao i njegovi prvi bajtovi, može se videti da izlaz *Greska_dužina* dobija aktivnu vrednost koja signalizira grešku u dužini prethodnog paketa. Pored toga, u istom taktu je prosleđen i signal *Kraj* za prethodni paket (u slučaju greške se postavlja vrednost signala *Kraj* koja označava najmanji mogući broj preostalih bajtova - jedan).

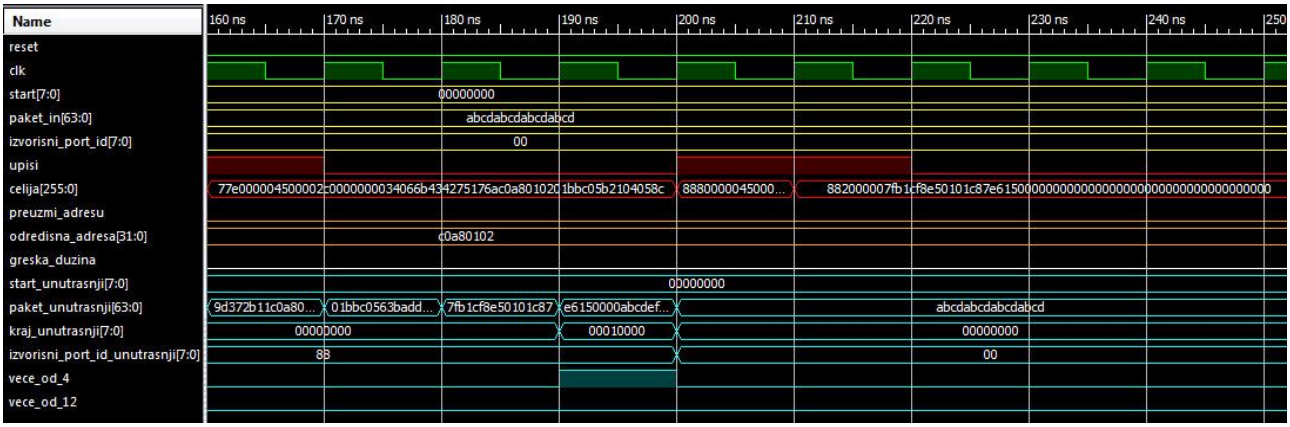
Zajedno sa poslednjim bajtovima drugog paketa postavljene su ispravne vrednosti signala *Kraj*, kao i *Vece_od_4* i *Vece_od_12*.



Slika 4.10 - Izgled prozora ISim simulatora, detekcija greške u dužini paketa- deo II

Na slici 4.11 se vidi da je od bajtova prvog paketa formirana samo jedna ćelija, i da ona u zaglavlju ima sva tri bita koji označavaju prvu ćeliju, neispravan paket i poslednju ćeliju postavljene na aktivne vrednosti.

Na izlazu su se pojavile i dve ćelije koje sadrže drugi IP paket, i kao što se može videti, njihova zaglavlja su pravilno formirana.



Slika 4.11 - Izgled prozora ISim simulatora, detekcija greške u dužini paketa- deo III

Funkcionalna simulacija je na prethodno opisan način izvršena i za sve ostale slučajeve. Dizajn je verifikovan pošto je potvrđen ispravan rad dizajna u svim test scenarijima.

4.2. Analiza performansi

Proces analize i sinteze izvršen je u ISE razvojnom okruženju za FPGA čipove proizvođača Xilinx. Odabran je čip XC5VFX70T koji pripada Virtex 5 familiji čipova. Dobijene informacije o zauzeću resursa i performansama su prikazane u tabeli 4.1.

Tabela 4.1 - Rezultati analize i sinteze.

Number of Slice Registers	1996 / 44800	(4%)
Number of Slice LUTs	2333 / 44800	(5%)
Number of occupied Slices	1450 / 11200	(12%)
Number of bonded IOBs	373 / 640	(58%)
Number of BUFGs	1 / 32	(3%)
Number of DSP48Es	1 / 128	(1%)
Maksimalna frekvencija (MHz)	157.828 MHz	

Na osnovu rezultata prikazanih u tabeli 4.1 može se videti da je postignuta zahtevana radna frekvencija dizajna od 156,25 MHz.

S obzirom da je postignuta radna frekvencija veća od 156,25 MHz, realizovani dizajn podržava 10G portove što je bio jedan od ciljeva implementacije. Upotrebom jačeg FPGA čipa mogle bi se podržati i više radne frekvencije. Takođe, važno je uočiti da su korišćeni skromni resursi što je veoma bitno jer na FPGA čip treba smestiti i blokove koji vrše druge funkcije paketskog procesiranja poput IP lukapa i raspoređivanja ćelija za slanje kroz paketski svič. Dizajn je realizovan da podrži bilo koje poravnanje IP paketa na nivou bajta u okviru 64-bitne magistrale podataka. Ukoliko bi, na primer, paketi počinjali uvek na najvišem bajtu 64-bitne magistrale podataka, realizovani dizajn bi se mogao dodatno uprostiti.

5. ZAKLJUČAK

U radu je prikazana implementacija ispitivanja validnosti IPv4 zaglavlja paketa, ekstrakcije odredišne IP adrese i segmentacije ispravnih paketa na ćelije. Realizovana implementacija može imati primenu kao prijemni deo paketskog procesora u Internet ruterima koji poseduju 10G portove.

Implementacija koristi veoma skromne hardverske resurse čipa, što je bitno jer na čipu treba da se smeste i druge funkcionalnosti paketskog procesora. Veoma je bitno naglasiti da nisu korišćeni resursi specifični za proizvođača (npr. interna memorija), pa je dizajn u potpunosti portabilan tj. može se prebaciti i na čipove drugih proizvođača poput Altere.

Prikazana implementacija omogućava fleksibilnost u izboru nižeg sloja prenosa, pošto niži sloj prilikom prosleđivanja paketa IP sloju ne mora da vodi računa o položaju početka i kraja paketa na unutrašnjoj magistrali širine osam bajtova. Ako bi niži sloj obezbeđivao da se početak paketa uvek prosleđuje u zasebnom ciklusu takta, tako da svih osam bajtova na magistrali pripadaju novom paketu, implementacija bi se mogla u velikoj meri uprostiti, i zauzeti manje resursa.

LITERATURA

[1] Wikipedia, *Internet Protocol* [Online]. Preuzeto sa: http://en.wikipedia.org/wiki/Internet_Protocol

[2] Marko Carević, Zoran Čiča, "FPGA Implementation of IP Packet Segmentation and Reassembly in Internet Router", *Serbian Journal of Electrical Engineering*, Vol. 6, No. 3: 399-407, December 2009.

[3] Wikipedia, *IPv4* [Online]. Preuzeto sa: http://en.wikipedia.org/wiki/IPv4

[4] Wikipedia, *IPv4 header checksum* [Online]. Preuzeto sa: http://en.wikipedia.org/wiki/IPv4_header_checksum

[5] H.J. Chao, C.H. Lam, E. Oki, *Broadband Packet Switching Technologies*, John Wiley and Sons , 2001.

[6] http://www.xilinx.com/

[7] https://www.wireshark.org/