Programska podrška za reprodukciju snimljenog video sadržaja na namjenskom uređaju

Pavlović, Ivan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PROGRAMSKA PODRŠKA ZA REPRODUKCIJU SNIMLJENOG VIDEO SADRŽAJA NA NAMJENSKOM UREĐAJU

Diplomski rad

Ivan Pavlović

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 20.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Ivan Pavlović
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 1078, 28.09.2017.
OIB studenta:	56372593518
Mentor:	Izv. prof. dr. sc. Marijan Herceg
Sumentor:	
Sumentor iz tvrtke:	Danijel Babić
Predsjednik Povjerenstva:	Doc.dr.sc. Ratko Grbić
Član Povjerenstva:	Izv. prof. dr. sc. Mario Vranješ
Naslov diplomskog rada:	Programska podrška za reprodukciju snimljenog video sadržaja na namjenskom uređaju
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Testiranje i verifikacija algoritama za pomoć vozaču predstavlja složen proces i zahtijeva vrijeme i resurse ukoliko se izvodi u stvarnom okruženju. Kako bi se testiranje i verifikacija ubrzali, okruženje se snima (video i drugi podaci) i kasnije reproducira u laboratorijima. U tu svrhu razvijaju se uređaji za snimanje i reprodukciju video sadržaja. U okviru ovog zadatka potrebno je omogućiti podršku za reprodukciju prethodno snimljenog video sadržaja na jednom takvom uređaju. Rješenje je potrebno realizirati na centralnoj procesorskoj jedinici Zynq XC7Z030, na ARM jezgri, u programskom jeziku C, gdje prethodno snimljeni video sadržaj treba dobavljati s računala preko PCIe
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	20.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2018.

Ime i prezime studenta:	Ivan Pavlović
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 1078, 28.09.2017.
Ephorus podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Programska podrška za reprodukciju snimljenog video sadržaja na namjenskom uređaju**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Marijan Herceg

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

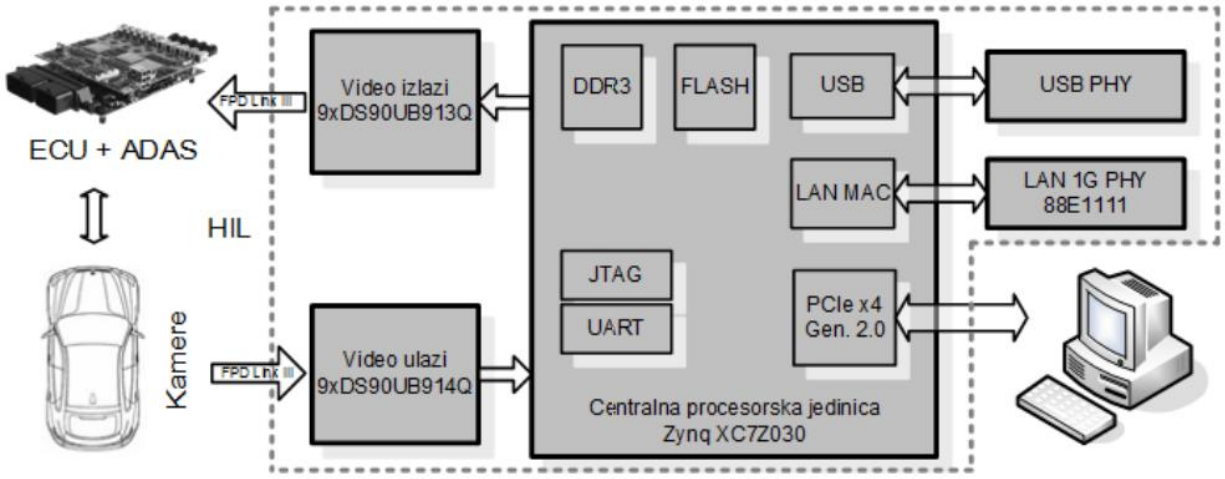
SADRŽAJ

1 UVOD

Jedna od glavnih tendencija razvoja tehnologije je pomoć čovjeku u njegovom svakodnevnom životu. Kako je putovanje automobilskim prijevozom postala čovjekova svakodnevica, pojavila se potreba za povećanjem sigurnosti i pomoći vozaču. Zbog tih potreba u automobilskoj industriji raste razvoj naprednih sustava za pomoć vozaču (engl. *Advanced driver-assistance systems*, ADAS). Ta tehnologija je skup računalnih sustava koji se nalaze unutar vozila kako bi olakšali proces vožnje i povećali sigurnost. Idejni začeci ADAS-a sežu od pojava pomoći vozaču u vožnji kao što su elektronička kontrola stabilnosti, sustav protiv blokiranja kotača, kontrola proklizavanja i slično. U današnje vrijeme najučestaliji primjeri ADAS tehnologija su prepoznavanje pješaka i prometnih znakova, rana detekcija mogućeg sudara, adaptivni tempomat i drugi. Glavna tendencija ADAS-a je da smanji pojavu ljudske greške prilikom vožnje koja može uzrokovati prometne nesreće. ADAS stoga može biti automatiziran i adaptivan te poboljšati sustave vožnje u kontekstu bolje vožnje. Uzimajući u obzir da je potrebno testirati svako softversko rješenje koje uključuje aspekt ljudske sigurnosti, povećava se potreba za zahtjevima takvih testiranja u laboratorijskim uvjetima. Implikacije čestih softverskih promjena u algoritmima se odražavaju na tome da se iste promjene testiraju na poligonu ili u realnim uvjetima u prometu. Laboratorijsko testiranje je rješenje tog problema koja eliminira troškove mogućih poligonskih i sličnih testiranja. U takvim laboratorijskim testiranjima potrebni su uređaji kao što su ADAS platforme koje imaju sposobnost skupiti video sadržaj koji okružuje vozilo koje se testira te ga reproducirati na drugom uređaju ili inteligentno obraditi skupljeni sadržaj. ADAS platforme su realizirane u obliku integriranih krugova na kojima se nalazi elektroničko sklopovlje koje može snimati okolinu ili obrađivati podatke u cilju pomoći vozaču u vožnji. Uz senzore koji snimaju okolinu kap što su kamere i senzori za mjerenje udaljenosti bitno je imati adekvatnu centralnu procesorsku jedinicu koja će skupiti i obraditi dane podatke. U trenutnom kontekstu autonomnosti vožnje možemo smatrati da se ADAS nalazi na trećoj razini ako postoji sljedećih pet;

a) Nulta razina - Nema automatizacije te vozač provodi sve operativne zadatke kao što su upravljanje, kočenje, ubrzavanje i slično.

b) Prva razina - Pomoć u vožnji gdje vozilo može pomoći u nekim aspektima poput dodatnog kočenja ali vozač i dalje upravlja sa svim zadatcima vožnje.
c) Druga razina - Djelomična automatizacija u kojemu vozilo može pomoći sa nekim funkcijama kao što su upravljanje ili ubrzavanje uz uvjet da vozač uvijek treba biti spreman preuzeti kontrolu nad vozilom.
d) Treća razina - Vozilo na ovoj razini promatra okolinu koristeći senzore te ima mogućnost automatskog kočenja u slučaju uočene opasnosti. Ljudska pažnja je i dalje kritična te čovjek treba biti u mogućnosti preuzeti kontrolu u svakom trenutku.
e) Četvrta razina - Na ovoj razini vozilo je u mogućnosti upravljanja, kočenja, ubrzavanja, promatranja okoline unutar koje može reagirati na događaje, zaobilaziti te koristiti signale. Na ovoj razini autonomni vozački sustav prvo obavještava vozača kada su uvjeti sigurni te tada vozač može preći u ovaj način vožnje.
f) Peta razina - Ova razina ne zahtjeva ljudsku pozornost. Nema potrebe za papučicama za ubrzanje ili kočenje kao ni volanom jer autonomno vozilo kontrolira sve zadatke, promatrajući okolinu te identificirajući jedinstvene vozačke uvjete kao što su zastoji u prometu.



Sl. 1.1 Topologija ADAS sustava [1]

ADAS platforma na kojoj je realiziran zadatak ovog diplomskog rada jest AMV Grabber (engl. *Automotive Machine Vision*). U topologiji ADAS sustava prikazanog na slici 1.1, AMV Grabber je moguće koristiti u dijelu koji opisuje skupljanje i reprodukciju video sadržaja. To je integrirana ploča čiji su primarni ciljevi dohvaćanje video sadržaja putem priključenih kamera i isporuka istih sadržaja na izlaz ploče koji se dalje može spojiti na druge integrirane ploče koje

imaju sposobnost inteligentno obraditi dani sadržaj. Spoj takvih ploča predstavlja okosnicu ADAS-a.

U poglavlju 2 biti će pružen opis AMV *Grabber*-a kao i opis korištenog razvojnog okružja i FreeRTOS-a. U poglavlju 3 se prikazuje testiranje *serializer*-a *loopback* programom i implementacija riješenja pomoću DMA (engl. *Direct Memory Access*). U poglavlju 4 su prikazani rezultati testiranja obavljenih tijekom izrade rada. Zaključak se nalazi u zadnjem poglavlju.

Kako bi se omogućila reprodukcija video sadržaja na namjenskom uređaju potrebno je osposobiti cijelu putanju prijenosa podataka od RAM-a računala preko PCIe sabirnice prema RAM-u ploče te od RAM-a prema izlaznom portu odnosno *serializer-u*. U programskom jeziku C u operativnom sustavu *FreeRTOS* je potrebno realizirati programske module koji će upravljati sklopovljem te inicijalizirati komponente kao što je PCIe sabirnica. Video sadržaj se prenosi na RAM putem DMA. Prilikom prijenosa podataka preko DMA koriste se *scatter-gather* liste koje omogućavaju kopiranje podataka sa memorije računala na memoriju ploče. U konvencionalnom prijenosu podataka podatci se prenose od ulaza do memorije preko CPU-a do memorije za što je potrebno 2 instrukcijska ciklusa. DMA radi na način da odvaja ulaz od CPU-a te direktno piše u memoriju za što je potrebno samo 1 ciklus. Konvencionalni načini prijenosa podataka zahtijevaju kontinuirane blokove memorije u koju bi skladištili podatke, a *scatter-gather* liste sastoje memorijske adrese memorijskih segmenata koji nisu u kontinuiranom redoslijedu. U ovom radu bilo je potrebno optimizirati DMA u *FreeRTOS*-u kako bi se ispravno prenosili *scatter-gather* blokovi podataka. Za vrijeme izrade rada je potrebno odabrati DMA koji će pravilno raditi sa spomenutim listama. Tijekom izrade rada se testiralo rješenje sa CDMA (engl. *Central Direct Memory Access*) i XDMA (engl. *Xilinx DMA*). Nakon omogućenog smještanja video sadržaja u RAM, ploče potrebno je konfigurirati VDMA (engl. Video Direct Memory Access) za slanje podataka na *serializer*. *Serializer* formatira podatke na način koji su pogodni za slanje, te se preko njega video sadržaj može poslati na vanjske uređaje ili pak nazad na samu ploču. Ako se video šalje nazad na samu ploču onda se kombinira spoj *serializer* i *deserializera* koji predstavljaju *loopback* funkcionalnost. *Loopback* funkcionalnost treba dati uvid u ispravnost izlaznih *port*-ova ploče, odnosno samih *serializer-a*. U radu će se koristiti:

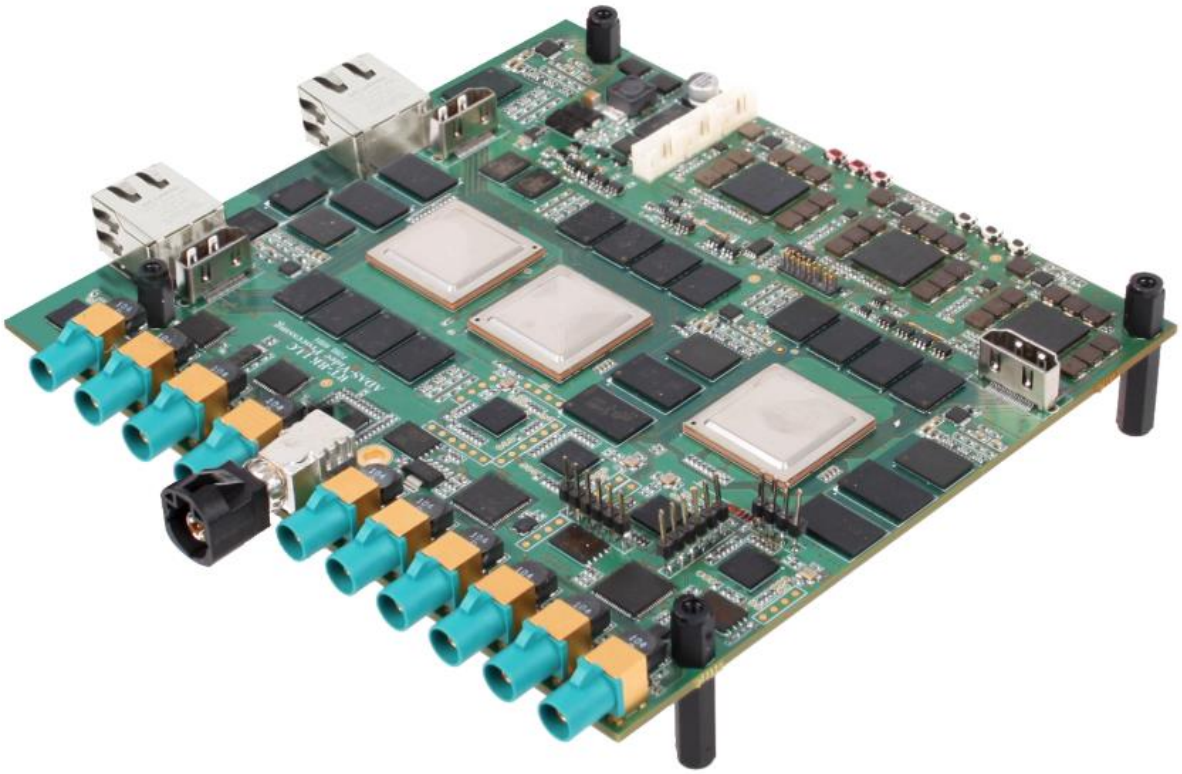
- Platforma: ADAS AMV *Grabber*
- Razvojno okružje: *Xilinx SDK* i *Vivado* 2016.2
- Programski jezik: C u *FreeRTOS*-u

- Operacijski sustav: *Microsoft Windows 10*

1.1 Pregled postojećih rješenja

1.1.1 AMV *Alpha board*

AMV *Alpha board* je kao i AMV *Grabber* napravljena na Institutu RT-RK Novi Sad. Njezina primarna namjena nije snimanje ili reprodukcija video sadržaja kao kod AMV *Grabber*-a iako sadrži video ulaze nego pokretanje ADAS algoritama koji se nalaze na njoj. ADAS algoritmi mogu upotpunjavati funkcije kao što je detekcija objekata od interesa u pruženom video toku. AMV *Alpha board* je moguće koristiti u sučelju sa AMV *Grabber*-om na način da AMV *Grabber* snima video sadržaj te ga reproducira na AMV *Alpha board* koji izvršava određene algoritme nad njim. Izgled AMV *Alpha board*-a je vidljiv na slici 1.2.



Sl. 1.2. Prikaz AMV *Alpha board*-a [2]

1.1.2 Xylon logiADAK *Automotive Driver Assistance* oprema

Xylon logiADAK Automotive Driver Assistance oprema je kao i AMV *Grabber* ADAS platforma temeljena na Xilinxovoj arhitekturi koja je razvijena za ADAS aplikacije koje

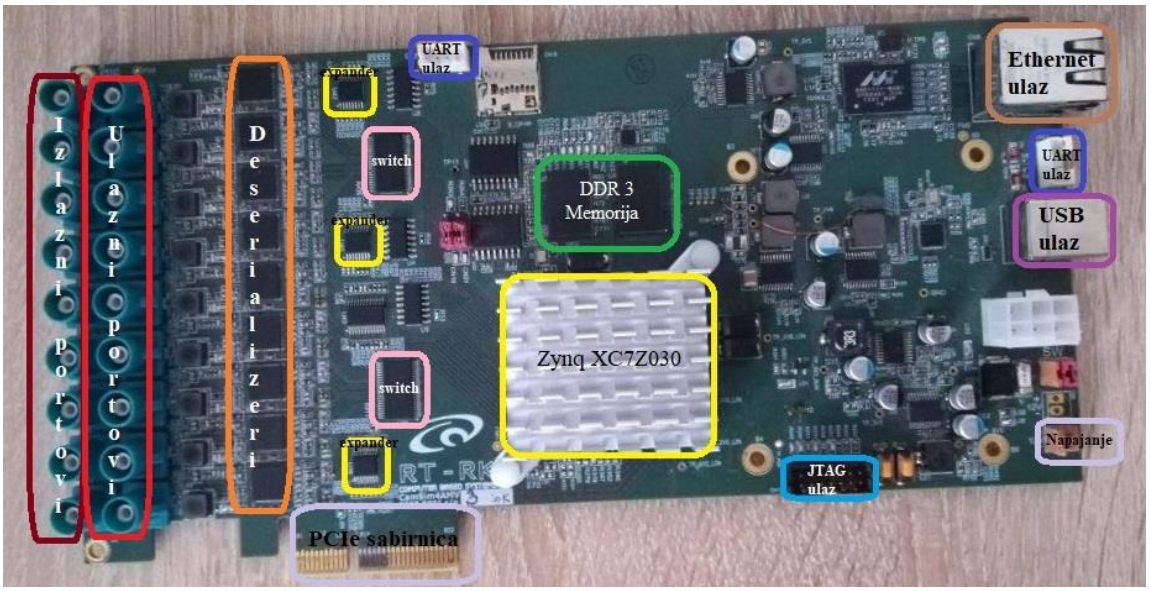
zahtjevaju procesiranje video sadržaja u stvarnom vremenu, paralelnu izvedbu kompleksnih algoritama i svestrano sučelje sa senzorima i komunikacijom sa samim vozilom u kojem je oprema integrirana. Xylon logiADAK *Automotive Driver Assistance* oprema nudi 360° okolni pogled, detekciju umora vozača, detekciju objekata u prometu te detekciju trake[3]. Prikaz opreme je vidljiv na slici 1.3.

Sl. 1.3. Prikaz Xylon logiADAK *Automotive Driver Assistance* opreme [3]

2 OPIS RAZVOJNOG OKRUŽENJA

2.1 Opis ploče AMV Grabber

AMV Grabber je platforma za snimanje video sadržaja i razvijena je na Institutu RT-RK Novi Sad. AMV Grabber je osmišljen kako bi u teoriji mogao snimati i reproducirati video sadržaje iz 9 izvora. Na njemu se u tu svrhu nalaze 9 ulaza i 9 izlaza. Oni su spojeni u paralelu, te tvore 9 parova *serializera/deserializera* [4]. *Serializer/Deserializer* je par funkcionalnih blokova koji se koriste pri širokopojasnom protoku podataka, odnosno u kontekstu rada to je prijenos širokopojasnog video sadržaja. *Serializer* (Sl. 2.3.) formatira podatke u format pogodan za slanje prema izlazu, a *deserializer* deserializira ulazni tok podataka kako bi on bio pogodniji za procesiranje. Deserializacija je proces koji uzima linearni tok podataka te ih izvodi u strukturalni oblik koji se može procesirati. CPU (engl. *Central processing unit*, centralna procesorska jedinica) na platformi je Zynq XC7Z030. CPU komunicira sa ulazno/izlaznim portovima (parovima *serializer/deserializer*) preko I^2C [5] (engl. *Inter-integrated Circuit*) protokola. I^2C protokol omogućava da više *slave* digitalnih integriranih krugova bude spojeno na jedan ili više *master* integriranih krugova. *Master-slave* je model komunikacije gdje jedan integrirani krug ima kontrolu nad više drugih integriranih krugova. Tako se na putanji prijenosa podataka od CPU do video izlaza, odnosno *serializer-a*, nalaze 2 *expander*-a I^2C i 3 *switch*-a. *Switch* kontrolira *ekspander-e* preko I^2C protokola koji dalje kontroliraju par *serializer/deserializer*. Expanderi putem I^2C protokola omogućuju paralelan rad ulaza i izlaza za određeni *port*-a. Paralelna izvedba ulaza i izlaza je vidljivi na lijevoj strani slike 2.2. Kako bi se osposobio pojedini par *serializer/deserializer* za korištenje u radu je bilo potrebno konfigurirati dane *switch-eve* i *ekspander-e* koji se nalaze na putanji do njega. CPU ima mogućnost dohvaćanja i slanja podataka preko PCIe sabirnice koja predstavlja most između AMV Grabbera i računala.



Sl 2.2. Prikaz komponenata AMV Grabber platforme sa gornje strane

Sl 2.3. Pogled na *serializere* na donjoj strani AMV Grabber platforme

CPU je temeljen na Xilinxovoj SoC arhitekturi. AMV Grabber ima dvojezgreni ARM® Cortex™-A9 procesor (*PS*, procesorski sustav) i 28 nm Xilinxovu programibilnu logiku (PL, programibilna logika). PL je logički uređaj koji služi za kreiranje konfigurabilnih digitalnih sklopova. Zynqov procesor je povezan sa PL-om preko AXI sabirnica, što znači da periferije koje su implementirane u PL-u mogu komunicirati sa softverom koji je programiran na procesoru. Programibilna logika je od iznimne važnosti u radu jer se pomoću nje programiraju DMA pomoću kojih se prenose video sadržaji. U kontekstu samoga hardvera, poluvodičke

uređaje koji su osnovani oko matrice konfigurabilnih logičkih blokova nazivamo FPGA [6] (engl. *Field Programmable Gate Arrays*). U tom kontekstu možemo smatrati PL i FPGA pojmovima koji označavaju istu funkcionalnost. FPGA ima mogućnost da bude programiran i reprogramiran kako bi obavljao određeni zadatak koji nije definiran prilikom same proizvodnje. To svojstvo ga razlikuje od ASIC (engl. *Application Specific Integrated Circuits*) odnosno integriranih krugova koji imaju specifičnu primjenu kao što je ARM procesor na koji je spojen.

Ploča ima 2 sučelja koja su korištena u radu za programiranje rada ploče sa strane računala. To su JTAG (engl. *Joint Test Action Group*) i UART (engl. *Universal Asynchronous Receiver-Transmiter*). JTAG (slika 2.4.) je hardversko sučelje i protokol koji se koristi za *debugging* ugradbenog hardvera koji se može sastojati od više mikroprocesora te kao što je korišten u ovome radu, za programiranje FPGA. *Debugging* je proces identifikacije i otklanjanja grešaka s namjenskog objekta, u ovom slučaju hardverskog dizajna. Za programiranje samog PS-a se koristi UART. UART (slika 2.5.) je hardverska periferija koja je memorijski mapirana i dostupna za upotrebu iz konteksta programa koji je pokrenut na mikroprocesoru (PS-u). UART može biti korišten kako bi poslao i primio podatke asinkrono bez određenog takta dok je JTAG sinkron te koristi određeni takt za komunikaciju.

Slika 2.4. JTAG sučelje

Slika 2.5. UART sučelje

FPD Link III (engl. *Flat Panel Display Link*) je komunikacijski kanal koji je korišten u izradi rada prilikom spajanja na ulazne ili izlazne *port*-ove AMV *Grabbera*. Izgled FPD Link-a vodiča je vidljiv na slici 2.6. FPD Link vodič omogućava prijenos HD digitalnih video podataka, kao i dvosmjerni kontrolni kanal, preko koaksijalnog vodiča. Na isti vodič je moguće spojiti kamere u slučaju kada bi bilo potrebe snimati video sadržaj sa AMV *Grabber*-om. FPD Link ugradbeni kontrolni kanal se koristi I^2C protokolom.

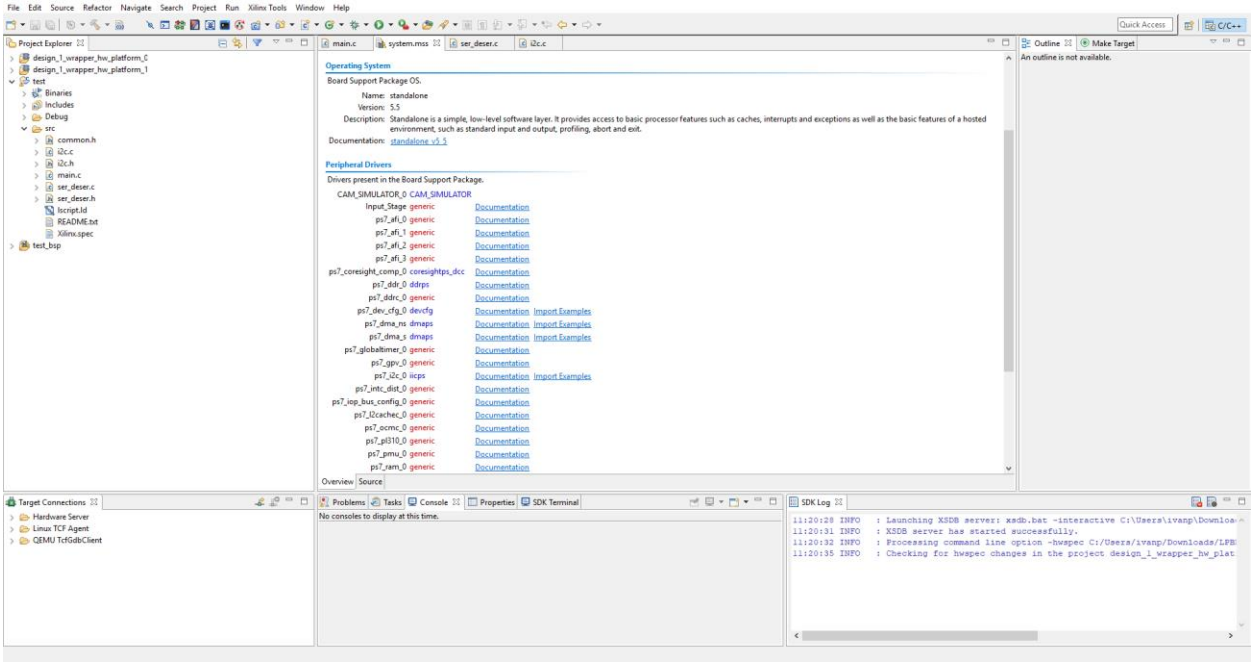
Sl 2.6 FPD Link III

2.2 Razvojno okružje i operativni sustav

Razvojno okružje je skup uputa, programskih knjižnica, relevantnih dokumentacija i sličnih alata koji omogućavaju korisnicima da u njima kreiraju svoja rješenja.

2.2.1 Xilinx SDK

Razvojno okružje korišteno za programiranje PS djela koda je *Xilinx SDK* [7]. U njemu je pisan C kod koji upravlja s radom samog dizajniranog hardvera. Izgled *Xilinx SDK* sučelja je vidljiv na slici 2.7. *XSDK* je integrirano razvojno okružje za kreiranje aplikacija na ugradbenim uređajima temeljenim na Xilinxovim mikroprocesorima.



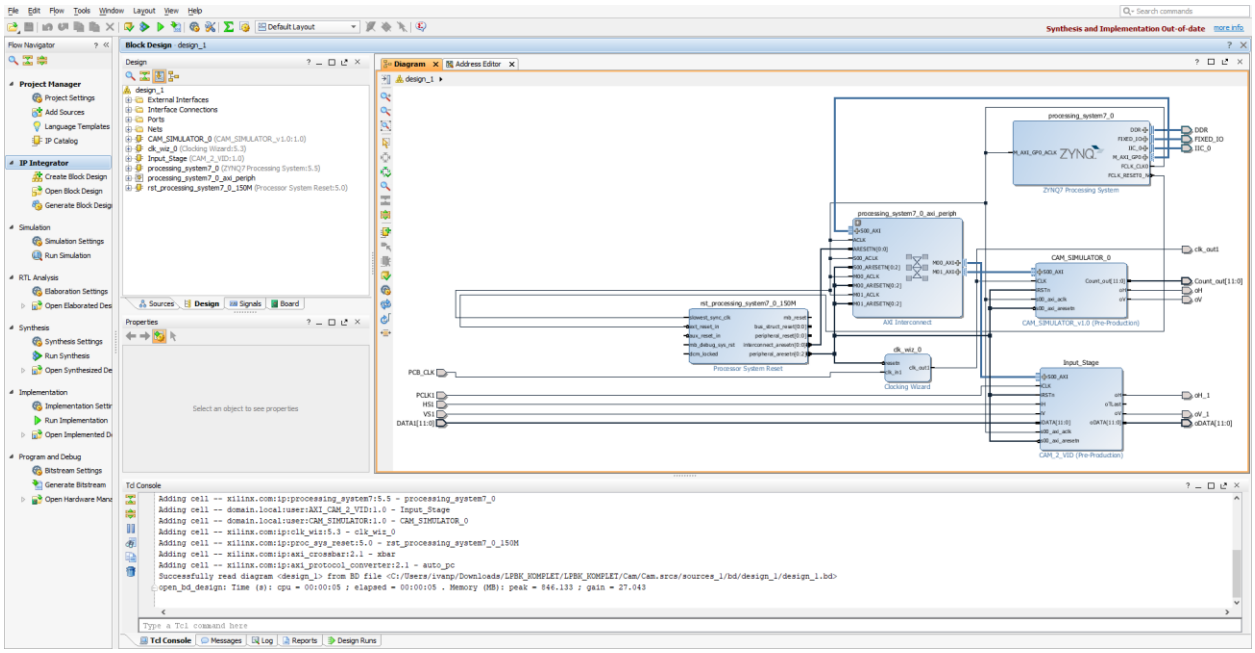
Sl 2.7 Izgled *Xilinx* SDK sučelja

2.2.2 Vivado 2016.2

Za programiranje rada FPGA korišten je *Xilinx Vivado* 2016.2 [8] koji omogućuje realizaciju određene funkcionalnosti koja nije predefinirana tijekom same proizvodnje sklopovlja. Izgled *Vivado* sučelja je prikazan na slici 2.8. Vivado nudi mogućnost optimizacije prilagođenih blokova, prijevoda HDL kodova, vremensku analizu, *debuggiranje* pojedinih blokova te simulaciju odziva na određene pobude. Određena funkcionalnost FPGA se ostvaruje slaganjem blokova u *Vivadu*. Ti blokovi se nazivaju IP (engl. *intelectual property*) jezgre.

Primjerice, za ovaj rad u poglavlju Prijenos podataka bilo je bitno optimizirati CDMA IP jezgru koja se smatra *soft* (hrv. mekanom) IP jezgrom jer ne postoji kao fizička manifestacija (engl. hard) nego kao hardverski deskriptivni kod (HDL) kod.

VHDL je programski jezik za opis digitalnih sustava koji se koristi za sintezu hardvera kao što je slučaj u ovome radu.



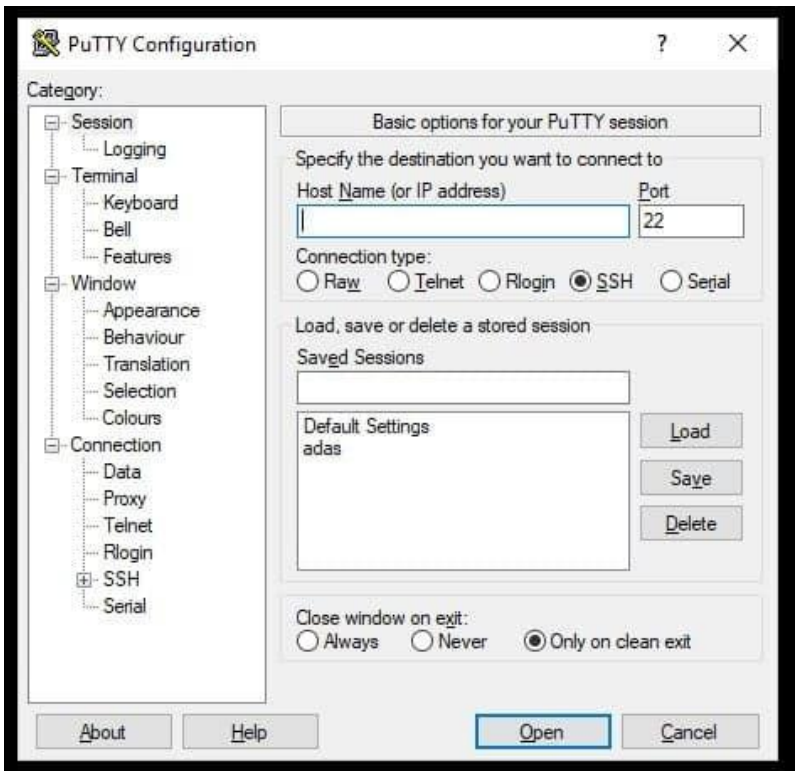
Sl. 2.8. Izgled *Vivado* sučelja

2.2.3 FreeRTOS

Operativni sustav koji se koristio u *XSDK*-u za razvoj rješenja je *FreeRTOS* [9] (engl. *Free Real Time Operating system*). To je operativni sustav u stvarnome vremenu koji je adaptiran za korištenje na mikroupravljačima kao što je Zynq koji je korišten u ovome radu. *FreeRTOS* predstavlja najnižu razinu apstrakcije između samoga softvera i hardvera što znači da ima sposobnost direktnom pristupu resursima sklopovlja. Apstrakcija u računarstvu označava proces isključivanja fizičkih, prostornih ili vremenskih atributa promatranog objekta kako bi se detaljnije promotrili detalji od interesa kao što su memorija. *FreeRTOS* je zasnovan na C programskom jeziku koji je jezik srednjeg sloja. To znači da teži nižim razinama apstrakcije kako bi direktno pristupio resursima same arhitekture sklopovlja.

2.2.4 *PuTTY*

PuTTY je besplatni emulator terminala koji je korišten prilikom testiranja ovoga rada. Na slici 2.9. je vidljivo kako je moguće postaviti tip veze. Za programiranje UART-om je korišten *serial* tip veze sa brzinom od 115200 b/s.

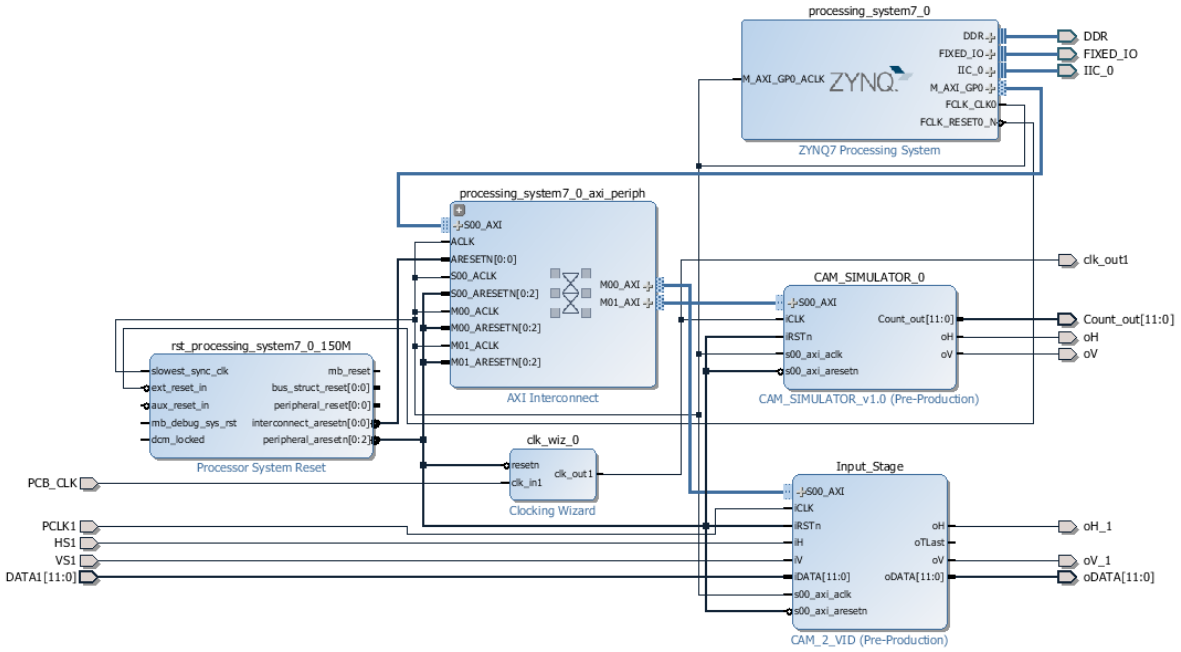


Sl 2.9. Izgled konfiguracijskih postavki PuTTY terminala

3 IMPLEMENTACIJA RIJEŠENJA

3.1 Loopback

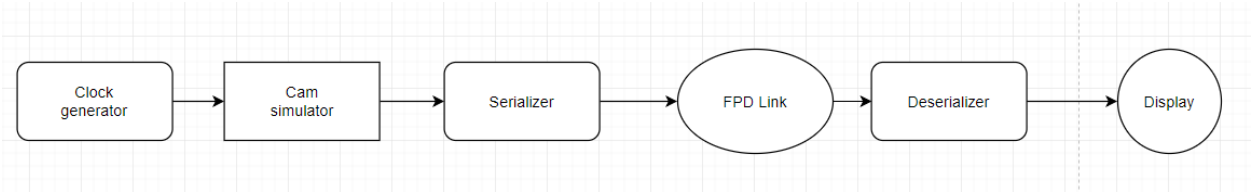
Loopback program je testni program koji provjerava ispravnost podataka koji putuju na putanji između *serializera* i *deserializera* preko FPD Link sučelja.



Sl 3.1. Blok dizajn *Loopback* programa

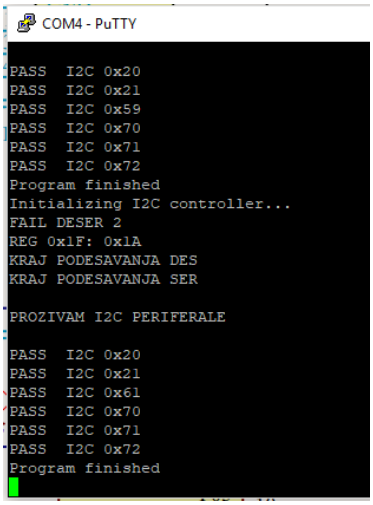
Na samoj ploči može se spojiti kamera na ulaz kako bi dobili video sadržaj sa izlaznog *port*-a ili možemo spojiti FPD Link na izlazni port kako bi poslali video sadržaj na izlaz AMV *Grabber*a. *Loopback* program omogućava da se izlaz ploče spoji na ulaz ploče koristeći FPD Link kako bi se testirala validnost podataka koji su generirani ili sama ispravnost komponenti kao što su *serializer*. Podatci se generiraju preko IP bloka pod nazivom CAM_SIMULATOR koji je vidljiv na slici 3.1. Navedeni blok generira video signal te ih šalje putem FPD Link sprege. Blok *Clocking wizard* mijenja frekvenciju na ulazu u blok na željenu frekvenciju za testiranje i održava ju stabilnom. Takt je potreban *serializer*-u i *deserializer*-u za rad te je to također takt na kojemu radi signal generator simuliranih podataka. Blok *ZYNQ7 Processing System* povezuje signalizaciju sa ostatkom sustava te procesira primljene podatke. Navedeni blok je povezan sa simulatorom i ulazom preko Axi Interconnect bloka koji djeluje kao sklopka u *master-slave* sučelju. Blok *Processing System Reset* pruža mehanizam za rukovanje uvjetima za resetiranje sustava. Na slici 3.2. je vidljiv dijagram toka podataka kroz *loopback* program. Blok

Generator takta (engl. *clock generator*) diktira stvaranje na izlazu te šalje dalje *serializer*-u koji sa *deserializer*-om stvara digitalno sučelje sa vrlo malim kašnjenjima.



Sl 3.2. Koncept i tok podataka kroz *loopback* program [10]

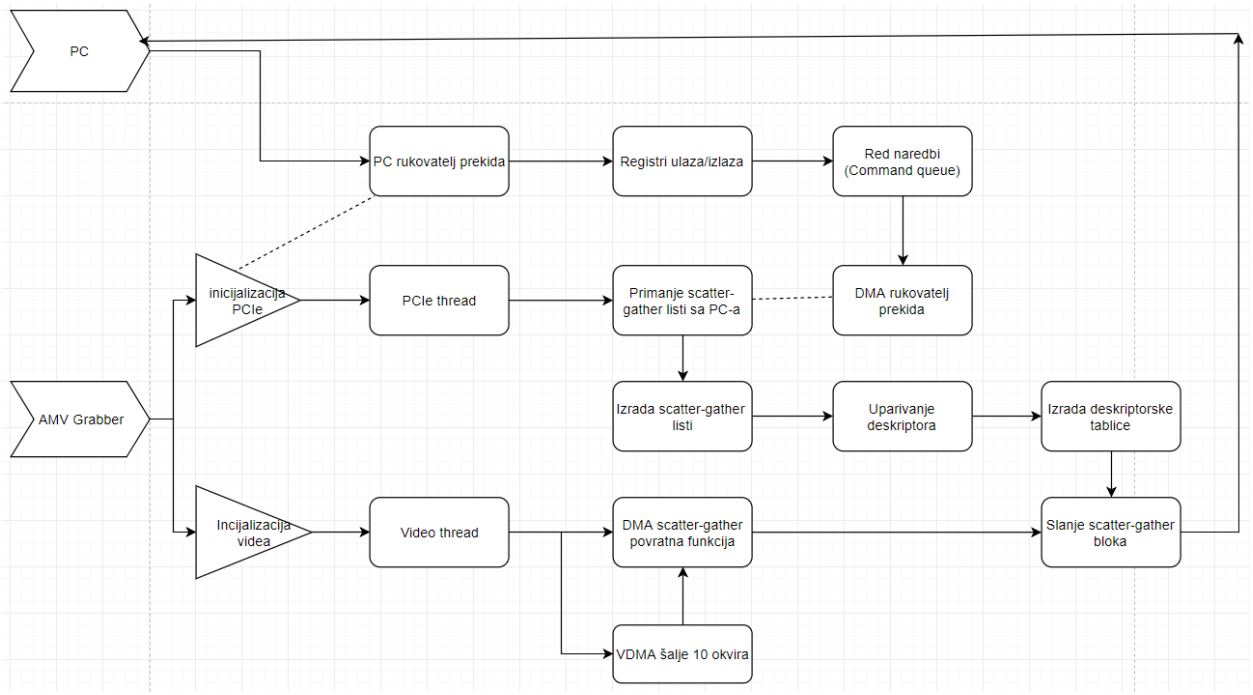
Loopback program je korišten kako bi se dijagnosticirala ispravnost samih *serializer-a* na ploči na kojoj se programiralo. Kako bi se provelo testiranje bilo je potrebno spojiti određeni *serializer* sa susjednim *deserializerom*, primjerice, *serializer* na portu 1 sa *deserializer-om* na portu 2. Kako bi se ustanovilo da je komponenta validna, PS je programiran da na terminalu ispiše prolaz za memorijski registar na kojemu se nalazi sama komponenta, 0x59 za *serializer* te 0x61 za *deserializer*.



Sl. 3.3. Ispis terminala tijekom provedbe *loopback* programa

Na slici 3.3 vidljiv je ispis s terminala u programu *PuTTY*. Program je programski izveden tako da inicira I^2C komunikaciju između PS-a i *expander* modula koji gasi sve *serializere* i *deserializere* te potom pali par *serializer*-a i *deserializer*-a koji se promatraju. Navedeni uređaji se pale pomoću I^2C protokola koji upisuje kontrolne podatke na osam bitne registre uređaja. Nakon paljenja se uređaji konfiguriraju istim I^2C protokolom. Poslije toga se prozivaju I^2C uređaji koji se nalaze na putanji do *serializer-a* kao što su sklopke te ako su uspješno konfigurirani odgovaraju sa „PASS“ porukom za određeni registar (npr, registar 0x20 i 0x21 je za sklopke) kao što je vidljivo na slici 3.3.

3.2 Smještanje podatka u memoriju (DMA)



Sl. 3.4. Dijagram toka stvaranja *scatter-gather* listi i slanja podataka

Kako bi se osiguralo smještanje podataka u memoriju AMV *Grabber*-a ili računala, prvobitno je potrebno pripremiti određenu količinu memorije na koja će se pisati. AMV *Grabber* je opremljen sa 1 GB DDR3 memorije. Prilikom prijenosa podataka na AMV *Grabber*, nemoguće je osigurati kontinuiran blok od jednog 1 GB te su zbog toga korištene *scatter-gather* liste za prijenos podataka. Izrada *scatter-gather* liste započinje kada AMV Grabber primi preko prekida (engl. *interrupt*) sa računala *scatter-gather* listu adresa koju dalje treba upariti sa svojim adresama. Prekidom se rukuje preko rukovatelja prekida koji komunicira sa inicijalizacijom PCIe prijenosa kao što je vidljivo na slici 3.4.

```
typedef struct {
    u32 next_desc;   /* 0x00 */
    u32 na1;         /* 0x04 */
    u32 src_addr;    /* 0x08 */
    u32 na2;         /* 0x0C */
    u32 dest_addr;   /* 0x10 */
    u32 na3;         /* 0x14 */
    u32 control;     /* 0x18 */
    u32 status;      /* 0x1C */
} __aligned(64) sg_desc_t;
```

Programski kod 3.1. Izgled strukture koja opisuje *scatter-gather* deskriptorsku tablicu

```
static sg_desc_t*       desc_table[3][2];
```

Programski kod 3.2 Postavljanje dimenzija deskriptorske tablice za prijenos 30 video okvira

Inače *scatter-gather* liste, koje se dobiju s računala mogu se koristiti direktno, no trenutna platforma ima određena ograničenja vezane za 64-bitno adresiranje preko PCIe. „AXI Memory Maped to PCI-Express“ IP jezgra ne dozvoljava direktno 64-bitno adresiranje. Umjesto toga zahtjeva pisanje gornjih bitova u posebni translacijski registar unutar jezgre te pisanje donjih bita na AXI sabirnicu. Kako bi se prešlo preko ovog ograničenja, za svaki unos u originalnoj *scatter-gather* listi, postoje dva unosa u posvojenoj *scatter-gather* listi. Prvi unos se naziva adresni deskriptor i on kopira samo gornje adresne bitove u translacijski registar. Drugi unos je podatkovni deskriptor koji zapravo kopira 4KB podataka preko PCIe sabirnice. Deskriptori koji se spominju u scatter-gather listama predstavljaju područje u memoriji koji pohranjuje atribute određene varijable. Lista *scatter-gather* adresa se nalazi na memorijskom prostoru od 4. do 64. MB na RAM-u ploče dok se sami video podatci nalaze od 64. MB do veličine koju mogu zauzeti dva video međuspremnika kao što je vidljivo u programskom kodu 3.3.

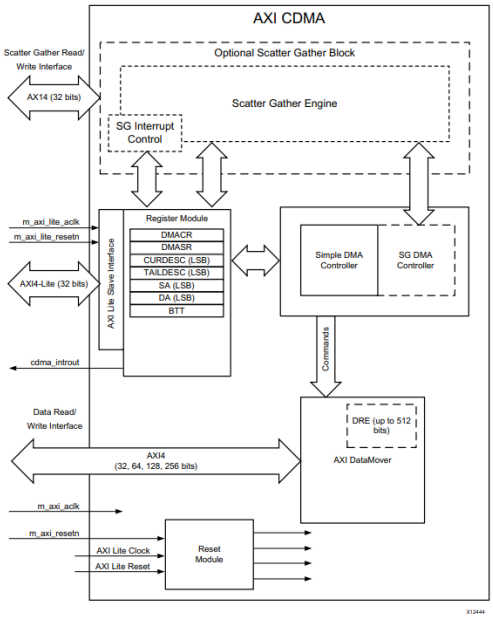
```
#define SG_LIST_ADDR            (RAM_BASE_ADDR + 0x00400000) /* RAM_BASE_ADDR + 4MB */
#define VIDEO_STORE_ADDR        (RAM_BASE_ADDR + 0x04000000) /* RAM_BASE_ADDR + 64MB */
```

Programski kod 3.3 Definicije memorijskih prostora za upis podataka na strani AMV *Grabber*-a

Nakon što su deskriptori AMV *Grabber*-a i PC-a upareni započinje stvaranje deskriptorske tablice čija je struktura vidljiva u programskom kodu 3.1. Deskriptorska tablica je realizirana kao dvodimenzionalni niz struktura dimenzija 3*2 kao što je vidljivo u programskom kodu 3.2. Ostvarena je na taj način kako bi mogla prenositi podatke za 30 okvira. Prva dimenzija označava da može imati 3 niza po jednom od dva moguća međuspremnika podataka, odnosno 3 niza od 10 okvira što predstavlja 30 okvira jedne sekunde jer je video namješten da radi u režimu od 30 okvira po sekundi. Nakon što je VDMA realizirala 10 okvira, DMA okida *scatter-gather* povratnu funkciju koja dalje okida slanje *scatter-gather* bloka podataka ka računalu.

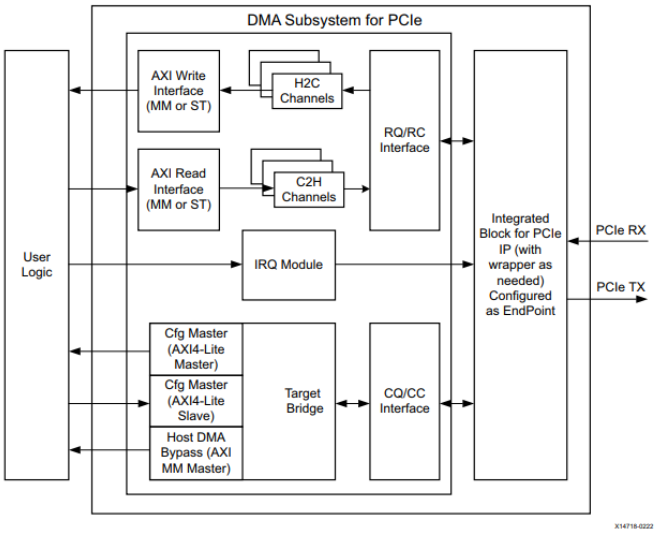
Prilikom izrade rada prvotno je korištena CDMA IP [11] jezgra u FPGA dizajnu. Detaljna shema bloka se vidi na slici 3.5. Ta jezgra ispravno prenosi podatke u smjeru prema računalu, no kada se zamjene podatkovni deskriptori koji označavaju izvorišnu i odredišnu

adresu kako bi smjer prijenosa podataka išao prema namjenskom uređaju, CDMA ne može ispravno obraditi podatkovni deskriptor zbog inherentne greške u FPGA dizajnu same jezgre.



Sl. 3.5. Opis AXI CDMA bloka [9]

Druga IP jezgra koja je korištena prilikom prijenosa prema namjenskom uređaju je DMA/Bridge Subsystem, također poznat pod imenom XDMA [12] pomoću kojega su uspješno prilikom testiranja na RAM AMV *Grabber*-a preneseni testni podatci. Detaljna shema komunikacije između korisnički postavljene logike i PCIe komponente se vidi na slici 3.6.

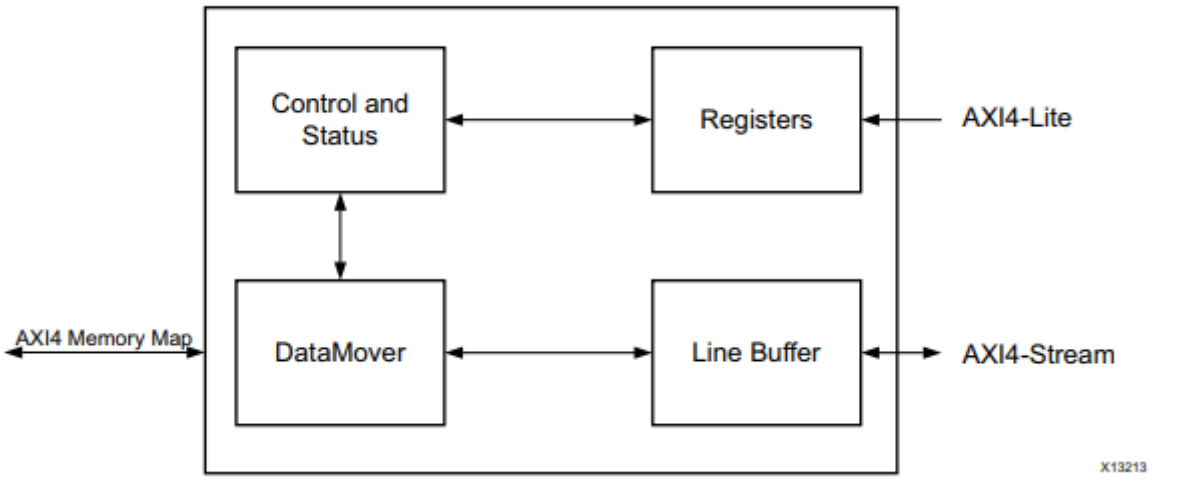


Sl. 3.6. Opis AXI XDMA bloka

3.3 VDMA

3.3.1 Konfiguracija VDMA IP bloka u FPGA dizajnu

Nakon što su podatci sigurno preneseni na RAM AMV *Grabber*-a potrebno je isprogramirati dizajnirani FPGA kako bi se omogućio prijenos podataka na izlaz AMV *Grabber*-a. Kreirani FPGA dizajn je vidljiv na slici 3.8**.** Krucijalni FPGA blok za prijenos video podataka koji se programira pri tome je VDMA. Generalni blok dijagram VDMA-e je vidljiv na slici 3.7**.** VDMA je dizajnirana da dozvoli efikasan pristup između toka podataka koji dolaze sa AXI4 toka i AXI4 sučelja kao što je naznačeno na slici 3.7**.** Prijenos podataka je omogućen tako što su registri programirani preko AXI4 sučelja te nadalje kontrolni i statusni logički blokovi generiraju prikladne naredbe *DataMoveru* kako bi inicirao naredbu čitanja sa AXI4 *master* sučelja. VDMA ima 2 smjera u kojima može raditi, to su smjer čitanja i pisanja. Ovaj rad je koristio smjer pisanja. U smjeru pisanja VDMA prihvaća okvire na AXI4 toku slave-a te ih zapisuje u memoriju sustava koristeći sučelje AXI4 *master*-a kao što je vidljivo na slici 3.7.

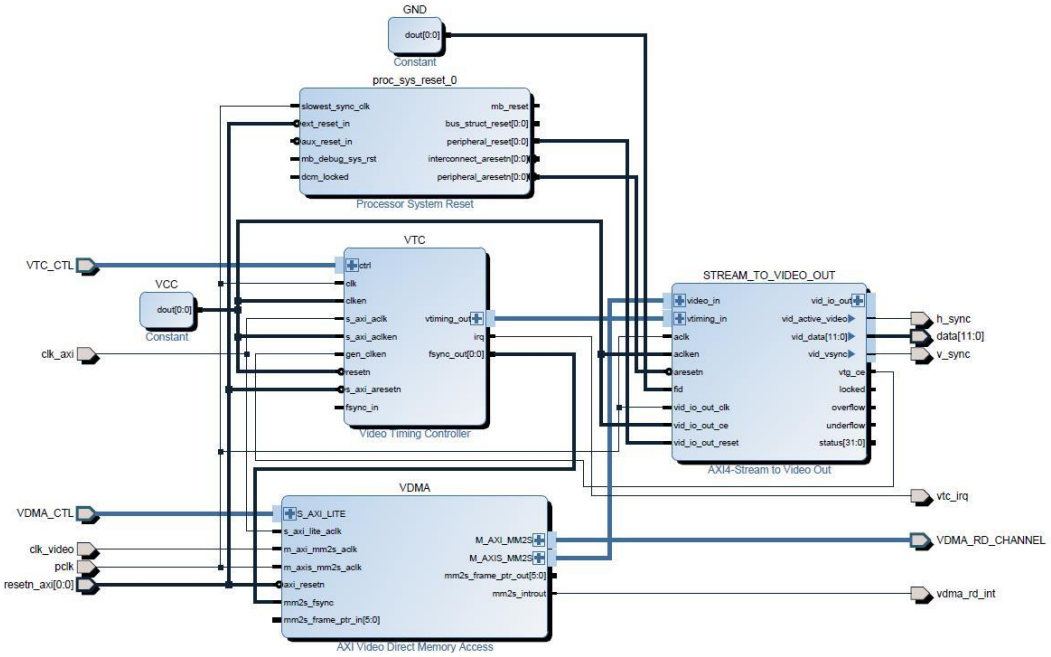


Sl. 3.7. Opis toka podataka kroz VDMA IP blok [13]

Na slici 3.8. je vidljivo kako su upareni blokovi VTC [14] (engl. *Video Timing Controller)* i VDMA. VTC služi za upravljanje vremenskih signala koji se koriste za sinkronizacijske procese. Bez VTC boka, sama VDMA ne bi mogla sinkronizirati parametre kao što su horizontalni i vertikalni sinkronizacijski pulsevi te aktivni video elementi slike. Na outputu je VTC dizajniran da generira horizontalne i vertikalne sinkronizacijske pulseve. VTC i CDMA su spojeni na ulaz bloka AXI4-stream to Video Out koji dalje šalje podatke na izlaz

AMV *Grabber*-a. Parametri koje šalje su podatci (engl. *data*) te horizontalna i vertikalna sinkronizacija (engl. *h-sync* i *v-sync*).

Prilikom testiranja je na dizajn dodan jedan ILA (engl. *Integrated Logic Analyzer*) IP blok koji djeluje kao *debugger* određenog sučelja. Podesivi ILA blok je logički analizator koji može biti korišten za promatranje unutarnjih signala u samom dizajnu.



Sl. 3.8. FPGA dizajn za prijenos ka izlazu AMV *Grabber*-a

3.3.2 Programska konfiguracija VDMA-e

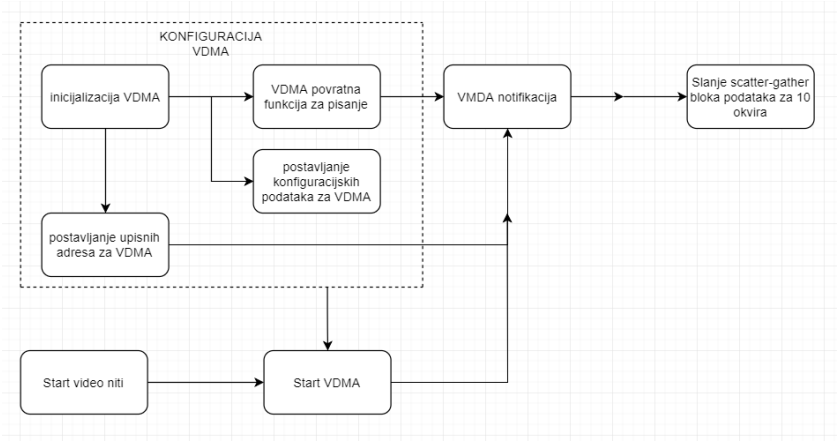
Kako bi se VDMA programski inicijalizirala potrebno ju je prvenstveno konfigurirati za njezinu namjenu. Programska podrška za programiranje VDMA-e dolazi u vidu biblioteka koje su pisane u C programskom jeziku te su dostupne od strane Xilinxa. Tako se u biblioteci xaxivdma.h nalazi struktura koja služi pri instanciranju *driver*-a za VDMA-u, kao što je vidljivo u programskom kodu 3.3.

```
typedef struct {
    UINTPTR BaseAddr;                   /**< Memory address for this device
*/
    int HasSG;                        /**< Whether hardware has SG engine */
    int IsReady;                      /**< Whether driver is initialized */

    int MaxNumFrames;                  /**< Number of frames to work on */
    int HasMm2S;                      /**< Whether hw build has read channel */
    int HasMm2SDRE;                   /**< Whether read channel has DRE */
    int HasS2Mm;                      /**< Whether hw build has write channel
*/
    int HasS2MmDRE;                   /**< Whether write channel has DRE */
    int EnableVIDParamRead;         /**< Read Enable for video parameters in
                             *  direct register mode */
    int UseFsync;                      /**< DMA operations synchronized to
                             * Frame Sync */
    int InternalGenLock;            /**< Internal Gen Lock */
    XAxiVdma_ChannelCallBack ReadCallBack;  /**< Call back for read channel
*/
    XAxiVdma_ChannelCallBack WriteCallBack; /**< Call back for write channel
*/

    XAxiVdma_Channel ReadChannel;  /**< Channel to read from memory */
    XAxiVdma_Channel WriteChannel; /**< Channel to write to memory */
      int AddrWidth;             /**< Address Width */
} XAxiVdma;
```

Programski kod 3.3. Struktura koja opisuje karakteristike VDMA-e

U programskoj realizaciji rada su korištene niti (engl. *threads*) [16]. One su ekonomičan način rada programa kod kojeg se nastoji što manje opteretiti određena jezgra procesora. Jedna nit je kontekst izvedbe programa, odnosno sve informacije koje su potrebne centralnoj procesorskoj jedinici da izvede niz instrukcija. Primjerice, u radu se koristi jedna nit za rad sa funkcijama koje rukovode VDMA-om, a druga za rad sa samim video sadržajem koji rukovodi samim video okvirima, te će oni preko prekida programa koristiti resurse centralne procesorske jedinice. Prekidi programa (engl. *interrupt*) [15] su signali koje procesor prima koji govore da određeni događaj treba više pažnje nego onaj događaj kojega procesor trenutno obrađuje. Povratna funkcija (engl. *Callback function*) je funkcija koja se može pozvati iz neke druge funkcije te je pozvana na kraju funkcije kojoj je proslijeđena.

Prilikom inicijalizacije VDMA-e postavljaju se dodatni parametri kao što je broj okvira koji se trebaju prenijeti. Kako je CDMA u prošlom poglavlju namješten da prenosi *scatter-gather* blokove za 10 okvira, tako se i instance VDMA konfiguracijskih struktura kao što je ona u programskom kodu 3.3. namještaju za prijenos u paketima po 10 okvira. Prilikom inicijalizacije VDMA-e je također potrebno postaviti povratne funkcije za pisanje koje će omogućiti uvid u rad same VDMA-e, tako da se postavlja povratna funkcija za ispis mogućih grešaka u prijenosu. Osim toga se postavlja povratna funkcija za pisanje koja poziva VDMA notifikaciju. Dotična notifikacija kao što je vidljivo u slici 3.9**.** Predaje *token* semaforu koji pušta funkciju iz CDMA programiranog u prethodnom poglavlju koja sada može poslati 10 okvira koje je VDMA prenijela.



Sl. 3.9. Dijagram konfiguracije VDMA-e i slanja podataka [10]

Rješenje je realizirano za slanje na jedan *serializer* tj. izlaz. U tom slučaju je promatran ILA IP blok koji je *debuggirao* podatke koji se šalju na izlaz kao što je prikazano u poglavlju „Rezultati testiranja“. Video sadržaj koji je kreiran za slanje na izlaz AMV *Grabber*-a je stvoren u obliku testnog uzorka.

4 Rezultati testiranja

4.1 Rezultati *loopback* testa

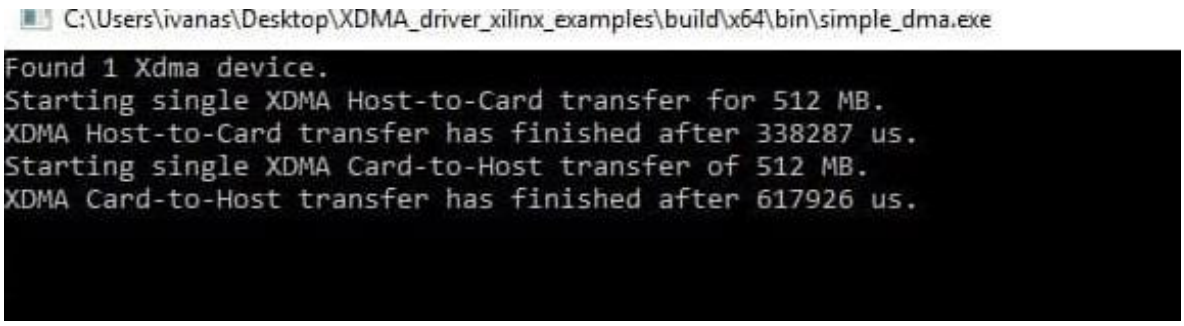
Testiranja koja su provedena tijekom izrade rada se odnose na testiranje ispravne konfiguracije samih *serializer*-a koja su vidljiva u tablici 5.1.

output(*serializer*)	Input (*deserializer*)	„0x59 i2c pass“	„0x61 i2c pass“
1	2	+	-
2	1	+	-
3	2	+	-
4	2	+	-
5	2	+	+
6	2	+	+
7	2	+	+
8	2	+	+
9	1	+	+

Tab. 5.1. Rezultati testiranja ispravnosti konfiguracije *serializer*-a

Kako bi podatci otišli na izlaz ploče preko jednog od 9 izlaznih *port-ova,* potrebno ih je prvotno prebaciti u serijski niz putem *serializer-a* koji hardverski prethodi samom izlaznom portu. Kako bi se testirala valjanost same hardverske komponente *serializer*-a, *loopback* test proziva I^2C adrese na kojima se nalaze *serializer*-i (0x59) i *deserializer*-i (0x61) te ispisuje one adrese sa koje dobije potvrdu. Ako se dobije potvrdna poruka za *serializer,* na terminalu se ispisuje poruka: „0x59 i2c pass“ . Red u Tab. 5.1. nazvan „0x59 i2c pass“ označava valjanost ispravne konfiguracije za *serializer* za dani broj *port*-a (od 1 do 9). Ako je serializer ispravno konfiguriran, to znači da preko njega možemo slati podatke na izlaz AMV *Grabber*-a. Red u Tab. 5.1. nazvan „0x61 i2c pass“ označava valjanost ispravne konfiguracije za *deserializer* za dani broj *port*-a (od 1 do 9). Ako je *deserializer* ispravno konfiguriran, to znači da preko njega možemo dobavljati podatke na ulaz AMV *Grabber*-a. Taj scenarij nije promatran jer nije od interesa u ovom radu, a uključen je u testiranje jer *serializer* hardverski dolazi u paru sa *deserializerom* [2].

4.2 Rezultati testiranja prijenosa s RAM-a računala preko PCIe sabirnice do RAM-a AMV *Grabber*-a koristeći XDMA



Sl. 4.1 Ispis terminala tijekom testiranja slanja podataka na RAM AMV *Grabber*-a

Prilikom slanja 512 MB testnih podataka u jednom od provedenih testova, sa Sl. 4.1. vidljivo je kako su podatci uspješno preneseni unutar 0.617 sekundi.

Tijekom testiranja slanja podatka preko PCIe sabirnice provedeno je mjerenje brzine prijenosa podataka u ovisnosti o veličini poslanih podataka i vremenu prijenosa, podatci su dani u tablici 5.2.

Veličina prenesenog sadržaja [MB]	Vrijeme potrebno za prijenos [μs]	Izračunata brzina prijenosa [*GB/s*]
100	110329	0,95
200	225483	0,93
300	337939	0,93
400	453106	0,93
500	546253	0,93
600	657483	0,95
700	759823	0,95

Tab. 5.2 Rezultati izračuna brzine prijenosa podataka na RAM AMV Grabbera preko PCIe sabirnice

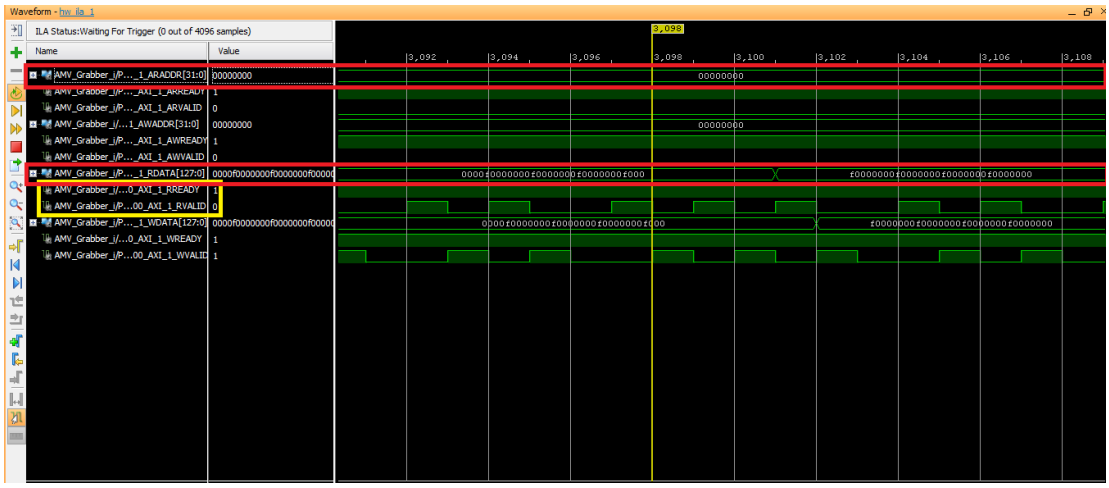
Brzina prijenosa se računala koristeći slijedeću formulu:

$$brzina\ prijenosa = \frac{\frac{veličina\ poslanog\ paketa\ [MB]*1024*1024}{vrijeme\ prijenosa\ [\mu s]*10^{-6}}}{10^9} \quad (4\text{-}1)$$

Sukladno s podatcima iz Tab. 5.2. vidljivo je kako brzina prijenosa podataka vrlo malo oscilira oko 0,95 *GB/s* te da nije ovisna o veličini samih podataka koji se prenose. Temeljeno na tim podatcima može se zaključiti da je AMV *Grabber* u mogućnosti prenositi video sadržaj sa svih 9 *port*-ova pod uvjetom da je sve ostalo programski ispravno realizirano.

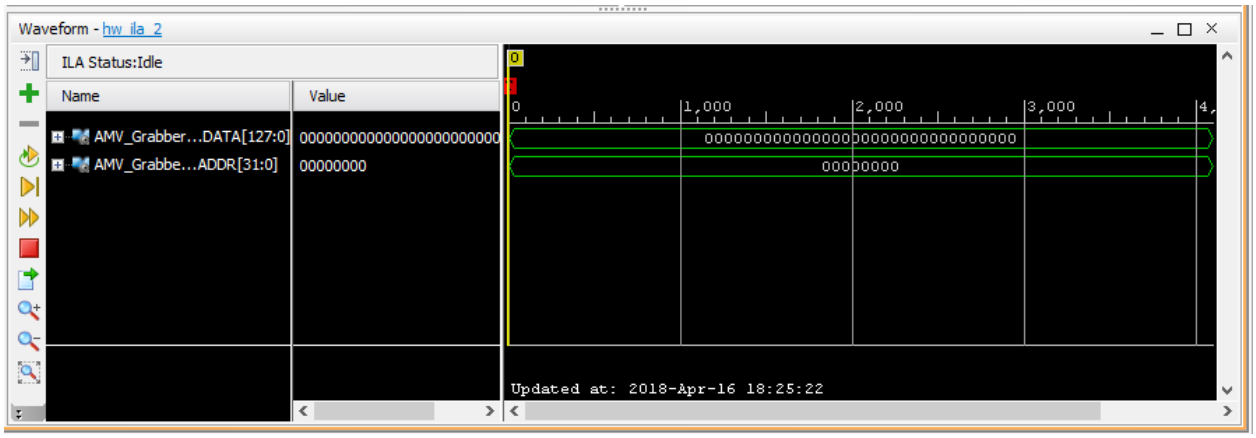
4.2.1 Rezultati testiranja prijenosa s RAM-a računala preko PCIe sabirnice do RAM-a AMV *Grabber*-a koristeći CDMA

Prvotna ideja prilikom implementacije rješenja je bila koristiti CDMA kao IP blok na kojega će se oslanjati prijenos podataka. Zbog nemogućnosti CDMA bloka da ispravno procesira 64-bitno adresiranje nije bilo moguće ostvariti prijenos podataka te se podatci nisu poslali na RAM AMV *Grabber*-a.



Sl. 4.2. Sadržaj ILA *debugger-a* na izlazu iz PCIe sabirnice, crvenim okvirom su označeni adresni i podatkovni podatci u smjeru čitanja (sa PC-a), a žutim okvirom sadržaj zastavica

Prilikom slanja testnih podataka ILA je namještena da promatra vrijednost podataka u smjeru čitanja i u smjeru pisanja, adrese za oba slučaja te popratne pomoćne zastavice. Na slici 4.2. je vidljivo kako vrijednost podataka u smjeru čitanja (RDATA) i u smjeru pisanja (WDATA) iznosi „f00000000“ što indicira da se podatci nisu poslali niti primili. Isto vrijedi i za promatrane adrese u smjeru čitanja (ARADDR) i u smjeru pisanja (AWADDR) koje imaju vrijednost „00000000“ a trebale bi imati konkretne adresne vrijednosti koje nisu jednake očitanoj. CDMA je dvosmjeran te ima mogućnost čitanja i pisanja. Zbog svoje interne konfiguracije CDMA brže obavlja proces čitanja nego pisanja. To će rezultirati nepravilnim redoslijedom obrade deskriptora. Do ove greške dolazi jer CDMA IP blok obrađuje podatkovni deskriptor prije nego obradi adresni, te zbog toga ne može upisati podatke na adekvatnu adresu. Budući da je CDMA obradio podatkovni adresor umjersto adresnog, u vremenu kada bi trebao obraditi adresni, on to ne učini zbog kašnjenja, pa adrese nemaju adekvatnu vrijednost. Dodatnu potvrdu pogrešnog ponašanja daje vrijednost zastavica koja je vidljiva u žutom okviru na slici 4.2. Na slici 4.2. je vidlijvo kako vrijednost zastavica RREADY (engl. *Read ready*) iznosi 1 a RVALID (engl. *Read valid*) iznosi 0. To znači da je ulaz spreman za čitanje ali nije validan



Sl. 4.3 Sadržaj ILA *debugger-a* koji prikazuje podatkovni i adresni sadržaj na ulazu u CDMA

ILA *debugger* prikazan na slici 4.3. promatra ulaz u sam CDMA proveden u jednom od mjerenja. Vidljivo je kako se očitane vrijednosti podudaraju sa onima na izlazu iz PCIe sabirnice. ILA promatra isti tok podataka kao i na slici 4.2. ali u ovom primjeru promatra samo podatkovne i adresne vrijednosti koje se odnose na one vrijednosti u crvenom okviru na slici 4.2. Po vrijednosti na izlazu iz PCIe sabirnice su jednake vrijednostima na ulazu u CDMA blok što govori da nema greške u samom komunikacijskom kanalu. Zbog navedenih rezultata je u daljnoj realizaciji rada korišten XDMA kako bi se ostvario prijenos podataka na RAM AMV *Grabber*-a.

4.3 Rezultat testiranja prijenosa podataka sa RAM-a AMV *Grabber*-a na izlaz

Sl. 5.2. Sadržaj ILA *debugger*-a na izlazu iz VDMA-e, crvenom bojom su označeni promatrani izlazi sa VDMA i VTC bloka

ILA *debugger* je spojen na data izlaz iz FPGA dizajna sa slike 3.8**.** ILA je konfigurirana na taj način da ima 15 okidača (engl. *trigger*) oko kojih se nalazi po 4000 segmenata podataka, odnosno okidač je pozicioniran na mjestu gdje bi se jedan okvir trebao nalaziti. Okidači su

označeni crvenim vertikalnim linijama na slici 5.2. Slika 5.2. u crvenim okvirima pokazuje kako su izlazi sa VDMA-e i VTC bloka (signal *active_video_out* i *VIDE..._vid_active_video*) sinkronizirani tek nakon četvrtog okidača, što pokazuje da su se od prvih deset okvira prenjeli samo drugih pet zbog sinkronizacije. Isti obrazac prijenosa se ponavlja za sve iduće pakete od 10 okvira što znači da predloženo rješenje ovoga rada prenosi 50% okvira u paketu. Do toga dolazi jer je potrebno određeno vrijeme kako bi se sinkronizirali VDMA i VTC blok prilikom početka rada. Kako bi se zaobišlo ovo ograničenje u inicijalnoj sinkronizaciji između VDMA i VTC blok, moguće je povećati zauzeće memorije za prijenos okvira. U slučaju prijenosa 10 okvira potrebno je zauzeti memoriju za 15 okvira, odnosno 5 okvira koji se izgube za sinkronizaciju i još 10 za prijenos korisnih okvira. Kada bi se programski realizirao prijenos u paketima od 20 okvira, tada bi se koristeći ovu metodu trebala zauzeti memorija za 25 okvira.

Generalno su opisi testiranja u poglavljima 4.2.1 i 4.3 nejasni. Nekako na slikama označi (drugom bojom ili nešto tako) što je što te kad taj dio spominješ u tekstu navedi kojom je bojom označen.

5 Zaključak

Namjera rada bila je osposobiti putanju prijenosa video sadržaja od RAM-a računala ka RAM-u AMV *Grabber*-a. Ostvaren je koncept rješenja koji funkcionira s video sadržajem testnog uzorka. Podatkovni sadržaj je uspješno prenesen u 2 dionice prijenosa. Prva je prijenos podataka sa RAM-a računala ka RAM-u AMV *Grabber*-a koji je ostvaren oslanjajući se na XDMA IP jezgru u FPGA dizajnu ploče što je opisano u poglavlju 3.2. Tijekom izrade tog djela bilo je potrebno optimizirati DMA programski u C programskom jeziku za dani FPGA dizajn. Problematika je nastala prilikom obrade podatkovnih deskriptora u slučaju korištenja CDMA IP bloka koji je bio prvotno korišten. Zbog prelaska na XDMA blok u FPGA dizajnu, većina testiranja je odrađena sa testnim video podatcima te se nije koristio video sadržaj koji je uhvaćen kamerama. Druga dionica prijenosa video sadržaja je prijenos testnog uzorka podataka koji je nezavisno kreiran na samom RAM-u AMV *Grabber*-a ka izlaznom *portu*, što je opisano u poglavlju 3.3. Tijekom izrade tog djela bilo je potrebno optimizirati FPGA dizajn koji se oslanja na VDMA IP bloku u C programskom jeziku. Prilikom testiranja solucije koja je predložena u tome poglavlju uvidjeli smo da se ispravno prenosi samo 50% video okvira.

Realizacija prijenosa video sadržaja na cijeloj putanji od RAM-a računala ka izlazu AMV *Grabber*-a se pokazala vrlo kompleksnom čak i kada se razdjeli na 2 dionice prijenosa. Programsku podršku je moguće nadograditi na taj način da se 2 dionice prijenosa integriraju u jednu koherentnu cjelinu te da se prenosi video sadržaj koji je uhvaćen sa kamera namjenskog uređaja. U tom slučaju bi AMV *Grabber* imao važnu ulogu u testiranju i verifikaciji algoritama za pomoć vozaču u ADAS industriji, odnosno u institutu RT-RK. U tom kontekstu bi se mogao upariti sa drugim ADAS platformama koje imaju sposobnost inteligentno obraditi dani sadržaj.

LITERATURA

[1] I. Krbanjević, I. Rešetar, V. Škobić; Univerzalna platforma za ispitivanje uređaja za mašinsku vizuelnu percepciju okoline u sistemima za pomoć u vožnji i autonomno kretanje vozila

Dostupno na: https://www.etran.rs/common/pages/proceedings/ETRAN2017/RT/IcETRAN2017_paper_RT1_4.pdf (pristup ostvaren 10.9.2018)

[2] „RT-RK - Automotive“

Dostupno na: http://www.rt-rk.com/services/automotive. (pristup ostvaren 15.9.2018.)

[3] Hardware Platforms - logiADAK

Dostupno na: https://www.logicbricks.com/products/logiadak.aspx (pristup ostvaren 15.9.2018.)

[4] Texas Instruments Datasheet, FPD LINK III SER/DES

Dostupno na : http://www.ti.com/product/DS90UB913Q-Q1 (pristup ostvaren 10.9.2018)

[5] Circuit Basics, Basics of the I2C Communication protocol

Dostupno na: http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ (pristup ostvaren 10.9.2018)

[6] Xilinx, What is an FPGA

Dostupno na: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html (pristup ostvaren 10.9.2018)

[7] Xilinx, Xilinx Software Development Kit

Dostupno na: https://www.xilinx.com/products/design-tools/embedded-software/sdk.html

(pristup ostvaren 10.9.2018)

[8] Xilinx, Vivado Design Suite HLx Editions - Accelerating High Level Design

Dostupno na: https://www.xilinx.com/products/design-tools/vivado.html

(pristup ostvaren 10.9.2018)

[9] FreeRTOS, Features Overview

Dostupno na: https://www.freertos.org/FreeRTOS_Features.html (pristup ostvaren 10.9.2018)

[10] Flowchart maker and online diagram software

Dostupno na: https://www.draw.io/ (pristup ostvaren 10.9.2018)

[11] Xilinx, AXI Central Direct Memory Access v4.1 LogiCORE IP Product Guide

Dostupno na: https://www.xilinx.com/support/documentation/ip_documentation/axi_cdma/v4_1/pg034-axi-cdma.pdf (pristup ostvaren 10.9.2018)

[12] Xilinx, DMA/Bridge Subsystem for PCI Express v4.0 Product Guide

Dostupno na: https://www.xilinx.com/support/documentation/ip_documentation/xdma/v4_0/pg195-pcie-dma.pdf (pristup ostvaren 10.9.2018)

[13] Xilinx, AXI Video Direct Memory Access v6.2 LogiCORE IP Product Guide

Dostupno na: https://www.xilinx.com/support/documentation/ip_documentation/axi_vdma/v6_2/pg020_axi_vdma.pdf (pristup ostvaren 10.9.2018)

[14] Xilinx, Video Timing Controller v6.1 LogiCORE IP Product Guide

Dostupno na: https://www.xilinx.com/support/documentation/ip_documentation/v_tc/v6_1/pg016_v_tc.pdf

(pristup ostvaren 10.9.2018)

[15] P. Romaniuk, Introduction to Multithreaded Programming in Embedded Systems

Dostupno na : https://www.elesoftrom.com.pl/en/os/multithreaded_programming.pdf

(pristup ostvaren 10.9.2018)

[16] EmbeddedCraft, Interrupt programming

Dostuopno na: http://www.embeddedcraft.org/interrupt.html (pristup ostvaren 10.9.2018)

Software for reproduction of captured video content on a dedicated device

ABSTRACT

This graduate thesis is based on enabling video data transfer from computer RAM to the output of the dedicated embedded device (AMV Grabber). CPU of the embedded device is based on the Xilinx SoC architecture. Software was written in C programming language in *FreeRTOS* and it deals with implementation of Xilinx IP DMA blocks such as VDMA or CDMA. The FPGA design of the board relies on these blocks. Software was realised in two main parts. First part was programming the appropriate DMA block dedicated for data transfer from computer RAM to AMV Grabbers RAM. Second part was realised by programming VDMA IP block to transfer data to the output of the AMV Grabber. The software solutions which is proposed in this graduate thesis has important implications in laboratory testings in ADAS industry.

Keywords: AMV Grabber**,** ADAS, DMA, embedded systems, C programming language, *FreeRTOS*

SAŽETAK

Ovaj rad se bavi realizacijom prijenosa video sadržaja sa RAM-a računala ka izlazu namjenskog ugradbenog uređaja (AMV *Grabber*). CPU namjenskog uređaja je temeljen na Xilinxovoj SoC arhitekturi. Programska podrška je realizirana u C programskom jeziku u *FreeRTOS*-u te se bavi korištenjem Xilinxovih IP DMA blokova kao što su VDMA ili CDMA na kojima se oslanja FPGA dizajn same ploče. Realizacija je ostvarena u dvije dionice podataka. Jedna je ostvarena programiranjem prikladnog DMA bloka za prijenos od strane računala ka RAM-u AMV *Grabber*-a. Druga je ostvarena programirajući VDMA da prenosi podatke na izlaz samog AMV *Grabber*-a. Rješenje kojeg predlaže rad za reprodukciju video materijala je važan čimbenik u okvirima laboratorijskog testiranja u ADAS industriji.

Ključne riječi: AMV *Grabber***,** ADAS, DMA, ugradbeni računalni sustavi, C programski jezik, FreeRTOS

ŽIVOTOPIS

Ivan Pavlović rođen je u Slavonskom Brodu 1.9.1993. Većinu života prebiva u Velikoj Kopanici gdje i pohađa osnovnu školu „Ivan Filipović“ koju prolazi s odličnim uspjehom. 2008. godine upisuje opću gimnaziju „Matija Mesić“ u Slavonskom Brodu koju 2012. završava s vrlo dobrim uspjehom. Iste godine upisuje preddiplomski studij elektrotehnike na Elektrotehničkom fakultetu sveučilišta Josipa Jurja Strossmayera u Osijeku. Na drugoj godini se opredjeljuje za smjer „Komunikacije i informatika“. Nastavlja diplomski studij 2016. na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, odabire izborni blok Mrežne tehnologije. Iste godine postaje stipendist instituta RT-RK u Osijeku, te slijedeće godine nastavlja stipendiranje u institutu gdje obavlja i praksu.

Ima odlično poznavanje engleskog jezika te je informatički pismen.

Potpis:

_____