#### ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU



## IMPLEMENTACIJA PROVERE ISPRAVNOSTI I AŽURIRANJA IPV4 ZAGLAVLJA UPOTREBOM *MicroBlaze* procesora

– Master rad –

Kandidat:

Srđan Durković 2013/3105

Mentor: doc. dr Zoran Čiča

Beograd, Decembar 2014.

# SADRŽAJ

SADRŽAJ	2
1. UVOD	
2. PROCESIRANJE PAKETA	4
3. MICROBLAZE PROCESOR	6
3.1. ARHITEKTURA PROCESORA	6
3.2. ORGANIZACIJA I PRINCIP RADA PROCESORA	7
3.3. PERFORMANSE PROCESORA	
4. EMBEDDED DEVELOPMENT KIT	
4.1. Base System Builder	
4.2. KREIRANJE NOVOG <i>EMBEDDED</i> PROJEKTA	
4.3. XILINX PLATFORM STUDIO	
4.3.1. Project Information Area	
4.3.2. System Assembly View	
4.4. PREGLED HARDVERSKE PLATFORME	
4.4.1. Razvoj hardverske platforme u Xilinx Platform Studio alatu	
4.4.2. Hardverska platforma u System Assembly View	
5. IMPLEMENTACIJA PROVERE I AŽURIRANJA IPV4 ZAGLAVLJA	
5.1. Create and Import Peripheral Vizard	
5.2. MODIFIKACIJA CIP TEMPLEJT FAJLA	
5.3. DODAVANJE KREIRANE PERIFERIJE U <i>PROCESSOR SYSTEM</i>	
5.3.1. Dodavanje periferije diode u projekat	
5.3.2. Eksportovanje dizajna i generisanje novog bitstream fajla	
6. VERIFIKACIJA DIZAJNA	57
7. ZAKLJUČAK	
LITERATURA	60

# 1. Uvod

Internet je jedan od simbola modernog doba. Krajem 20-og i u 21-om veku, Internet tehnologije su doživele vrtoglav razvoj. Iako je prvobitno razvijan za potrebe američke vojske, razvojem tehnologije i povećanjem dostupnosti personalnih računara, Internet je postao dostupan većem broju ljudi. To je uslovilo i razvoj novih servisa, koji su privlačili nove korisnike, što je dalje uslovljavalo razvoj Internet infrastrukture. Ključna stvar po kojoj se Internet razlikovao od dotadašnjih komunikacionih tehnologija (npr. telefonije ili telegrafije) je to što Internet koristi komutaciju paketa. To je zahtevalo i razvoj novih mrežnih uređaja (rutera) koji će upravljati tim paketima. Poruke se pre slanja dele na pakete, i zatim nezavisno prenose kroz mrežu. Zadatak rutera je da na osnovu informacija iz zaglavlja paketa, prosledi paket na odgovarajući izlaz. Kako je saobraćaj tokom godina eksponencijalno rastao, to su i kapaciteti mrežnih uređaja postali kritični. FPGA čipovi predstavljaju vrlo zgodan izbor za razvoj novih i unapređenje postojećih tipova rutera, pošto omogućavaju laku modifikaciju i unapređenje dizajna, kao i dodavanje novih funkcionalnosti.

Tema ovog rada je upravo implementacija jedne funkcije paketskog procesiranja (provera i ažuriranje IPv4 zaglavlja) upotrebom *MicroBlaze soft* procesora. Za realizaciju sistema je korišćena *ML507* razvojna ploča proizvođača *Xilinx*. Odabran je *soft* tip procesora, jer pruža veliku fleksibilnost u pogledu izbora periferija i performansi samog procesora. Za razvoj ovog dizajna korišćen je softverski alat EDK (*Embedded Development Kit*) proizvođača *Xilinx*. EDK se sastoji od dve komponente: XPS (*Xilinx Platform Studio*) i SDK (*Software Development Kit*). XPS služi za razvoj hardverskog dela projekta, i znatno olakšava dizajniranje sistema. Za razvoj softverskog dela projekta, korišćen je SDK, koji je kompatibilan sa EDK alatom. U radu je opisan tok razvoja dizajna, sa objašnjenjima pojedinih opcija koje se nude korisnicima. EDK pruža veliku fleksibilnost u pogledu menjanja dizajna i dodavanja novih funkcionalnosti. SDK je razvojno okruženje u kome se vrši razvoj aplikacije u C/C++ programskim jezicima, koje procesor izvršava. U ovom radu, EDK je korišćen za kreiranje hardverske osnove projekta i razvoj nove periferije, dok je u SDK alatu razvijena aplikacija koja vrši proveru i ažuriranje IPv4 zaglavlja.

Rad je organizovan na sledeći način. U drugom poglavlju je objašenjeno šta se sve podrazumeva pod paketskim procesiranjem, i koje se to funkcije izvršavaju u ruterima. Jedna od njih je tema ovog rada. U trećem poglavlju je dat pregled arhitekture *MicroBlaze* procesora i izložene su njegove najznačajnije karakteristike. U četvrtom poglavlju su objašnjene mogućnosti koje EDK alat pruža korisnicima, kao i njegova uloga u razvoju *embedded* sistema. Predstavljen je proces razvoja dizajna, kao i opcije koje su na raspologanju korisnicima. U petom poglavlju se nastavlja opis razvoja dizajna, implementacijom periferije koja služi za proveru i ažuriranje IPv4 zaglavlja. Hardverski deo periferije se odnosi na ulazno/izlazne jedinice i registre za skladištenje podataka, dok se u softverskom delu razvija aplikacija koja upravlja tom periferijom. U šestom poglavlju je izvršena verifikacija rada periferije, a dat je i tabelarni pregled zahtevanih resursa koje zahteva ovaj dizajn. Na kraju je dat kratak zaključak u kome su izloženi rezultati rada i mogućnosti za dalje razvijanje ovog projekta.

# **2.** PROCESIRANJE PAKETA

Internet je danas dominantna telekomunikaciona mreža. Pored prvobitnih servisa poput pretraživanja veba, razmene tekstualnih poruka ili fajlova, Internet nam danas nudi i druge mogućnosti poput prenosa govora, TV signala i slično. Dakle, Internet infrastruktura pruža mogućnosti za implementaciju više različitih servisa. Stoga, može se reći da je danas pristup Internetu osnovna potreba čoveka bilo da sa radi o zadovoljenju poslovnih, školskih ili privatnih obaveza. Kako je Internet globalna telekomunikaciona mreža nastala povezivanjem više različitih mreža, njegov uspeh bi bio nemoguć bez uvođenja jedinstvenog standarda. Iako je IETF (*Internet Engineering Task Force*) definisao OSI referentni model, Internet se danas de fakto zasniva na TCP/IP (*Transport Control Protocol/Internet Protocol*) protokolu, prema kome razlikujemo 5 različitih slojeva: fizički, MAC, mrežni, transportni i sloj aplikacije.

IP tehnologija se zasniva na komutaciji paketa. To znači da se na predajnoj strani podaci segmentiraju na pakete koji se dalje kroz mrežu prenose nezavisno. Na prijemu se vrši desegmentacija odnosno rekontrukcija primljenih paketa. Svaki paket se sastoji od dva dela:

- Payload, odnosno korisna informacija koja se prenosi
- Zaglavlje, odnosno informacije koje služe za pravilno prosleđivanje i upravljanje paketima

Zaglavlje se sastoji od više delova jer svaki sloj ubacuje svoje informacije. Osnovu Interneta čine ruteri čiji je zadatak da provere ispravnost paketa, odrede rutu kojom može da stigne do svog odredišta i proslede ga na odgovarajući izlazni port. Dakle, u ruterima se vrši obrada paketa i to na više nivoa. Procesor fizičkog sloja prima signal, izdvaja takt iz njega, uobličava signal, usaglašava fazu signala, vrši detekciju i dekodovanje bita. Na nivou linka vrše se sledeće funkcije: razgraničavanje okvira, izdvajanje adrese odredišta i prijem okvira ako se adresa odredišta poklapa sa lokalnom adresom, provera ispravnosti okvira, određivanje mrežnog protokola [1].

Proces rutiranja ili prosleđivanja paketa predstavlja najsloženiju mrežnu funkciju, naročito ako se ima u vidu da su brzine prenosa podataka eksponencijalno rasle poslednjih godina. Najprostije, zadatak rutera je sledeći:

- Provera da li je primljeni paket validan
- Određivanje na koji izlazni port treba proslediti paket (lukap funkcija)
- Ažuriranje zaglavlja paketa
- Prosleđivanje paketa na odgovarajući izlazni port

Za izvršavanje ovih funkcija ruter koristi informacije iz IP zaglavlja. Ukoliko se utvrdi greška u zaglavlju, paket se odbacuje. Kako bi se performanse rutera optimizovale, a i zbog mnoštva funkcionalnosti koje poseduju, ruteri se dele na više ravni. To su:

- Ravan podataka
- Kontrolna ravan

• Upravljačka ravan

Ravan podataka je podsistem mrežnog čvora koji prima i šalje pakete od i ka interfejsima, obrađuje ih u skladu sa implementiranim protokolom i dalje ih prosleđuje ili odbacuje. Ove funkcije se implementiraju u hardveru.

Kontrolna ravan obuhvata širok skup kontrolnih protokola koji obezbeđuju ispravno i kvalitetno obavljanje osnovne funkcije rutera. To se odnosi na mehanizme i funkcije kontrole rada ravni podataka poput protokola rutiranja, mehanizama kvaliteta servisa, rezervacije resursa, multikast podrške i sl. Ove funkcije zahtevaju razmenu kontrolnih poruka koje zauzimaju značajno manje resursa nego podaci koji se prenose. Implementiranje ovih protokola u ravni podataka bi dovelo do nefleksibilnosti u konfigurisanju rada rutera kao i prevelikih hardverskih zahteva i one se implementiraju softverski.

Upravljačka ravan obuhvata funkcije kao što su: konfigurisanje i nadgledanje rada rutera, alarmiranje u slučaju neispravnosti, omogućavanje udaljenog pristupa ruteru i sl. Funkcije upravljačke i kontrolne ravni se izvršavaju u okviru istog operativnog sistema, tako da se često ove dve ravni posmatraju kao jedna.

Ravan podataka	Kontrolna ravan	Upravljačka ravan
Provera ispravnosti paketa, ispitivanje i modifikacija IP zaglavlja, filtriranje i klasifikacija paketa, lukap funkcija, šifrovanje/dešifrovanje, mehanizmi odbacivanja paketa u slučaju zagušenja, segmentacija paketa na ćelije fiksne dužine, baferisanje ćelija, raspoređivanje i prosleđivanje ćelija sa ulaznih na izlazne porove itd.	Protokoli rutiranja i određivanje lukap tabele (OSPF, RIP, BGP, MPLS, LDP), multikast podrška (IGMP, PIM), rezervacija resursa (RSVP), autentifikacija i autorizacija, menadžment ključeva za šifrovanje, kontrola pristupa itd.	Nadgledanje i dijagnostika (SNMP, ICMP), udaljeni pristup, interfejs za konfigurisanje (CLI)

U tabeli 2.1 je dat pregled funkcija koje izvršavaju u odgovarajućim ravnima.

Tabela 2.1. Funkcije koje vrše kontrolna, upravljačka i ravan podataka[2]

# **3.** *MicroBlaze* procesor

*Embedded* sistemi su računarski sistemi posebne namene. Za razliku od računara opšte namene ovi sistemi izvršavaju specifične, prethodno definisane zadatke i zato su vrlo podesni za obavljanje nekih funkcija u mrežnim uređajima. *Xilinx embedded* rešenja koriste dva tipa procesora: *MicroBlaze* i *PowerPC*. *MicroBlaze* procesor spada u kategoriju tzv. *soft core* procesora, što znači da koristi opšte resurse FPGA čipa pa je moguće kreirati raznovrsne konfiguracije, a samim tim je i pogodan za obavljanje različitih funkcija. Može da se koristi na bilo kojoj *Xilinx* FPGA platformi koja ima dovoljno opštih hardverskih resursa.

#### 3.1. Arhitektura procesora

*MicroBlaze* procesor je 32-bitni RISC (*Reduced Instruction Set Computer*) tip procesora koji može dostići brznu i do 3GHz na najnovijim uređajima. Osnovne karakteristike ovog procesora su[3]:

- Trideset dva 32-bitna registra
- 32-bitne instrukcije
- Odvojene 32-bitne magistrale za instrukcije i podatke koje odgovaraju OPB (*On-chip Peripheral Bus*) specifikaciji
- Odvojene 32-bitne magistrale za instrukcije i podatke sa direktnom konekcijom na *on-chip block* RAM preko LMB (*Local Memory Bus*)
- 32-bitna adresna magistrala

Ovom osnovnom dizajnu mogu biti dodate neke naprednije karakteristike da bi se zadovoljile potrebe konkretne *embedded* aplikacije kao što su[3]:

- Keš memorija za instrukcije i podatke
- Hardverska logika za debagovanje
- FSL (*Fast Simplex Link*) podrška
- Hardverski množač
- Floating Point Unit

Ova fleksibilnost dozvoljava korisniku da balansira između zahtevanih performansi konkretne aplikacije i zahteva za hardverskm resursima *soft* procesora. Slika 2.1 prikazuje izgled *MicroBlaze* sistema. Delovi u belom su kičma *MicroBlaze* arhitekture dok su osenčeni delovi opcione mogućnosti dostupne zavisno od potreba konkretne *embedded* aplikacije. Kako je *MicroBlaze soft-core* mikroprocesor, opcione mogućnosti koje nisu implementirane neće zauzimati FPGA resurse [4].



Slika 2.1. MicroBlaze procesor [4]

*MicroBlaze pipeline* je paralelni *pipeline*, podeljen u tri staze: *fetch*, *decode* i *execute*. Generalno, svaka staza zazima jedan takt da se kompletira. Konsekventno, potrebna su tri takta (ne računajući kašnjenja) da bi se instrukcija izvršila. Svaka faza je aktivna jedan takt, tako da se tri instrukcije mogu izvršavati simultano, svaka na jednoj od tri *pipeline* staze. *MicroBlaze* implementira *Insruction Prefetch Buffer* koji smanjuje uticaj kašnjenja *multi-cycle* instrukcija. Dok je *pipeline* zauzet *multi-cycle* instrukcijama, *Instruction Prefetch Buffer* nastavlja da učitava sekvencijalne instrukcije. Jednom kada *pipeline* nastavi izvršavanje, *fetch* staza može učitati novu instrukciju direktno iz *Instruction Prefetch Buffer* umesto da čeka pristup memoriji instrukcija. *IPB* je deo kičme *MicroBlaze* arhitekture i nije isto što i keš memorija za instrukcije [4].

### 3.2. Organizacija i princip rada procesora

*MicroBlaze* procesor koristi *bigendian* format za zapisivanje podataka. Hardverski podržani tipovi podataka za *MicroBlaze* su reč, pola reči i bajt. Organizacija bita i bajtova za svaki tip je data u tabelama 2.2-2.4 [3].

Big-Endian Byte Adress	n	n+1	n+2	n+3
Big-Endian Byte Significance	MSByte			LSByte
Big-Endian Byte Order	n	n+1	n+2	n+3
Big-Endian Byte-Reversed Order	n+3	n+2	n+1	n
Little-Endian Byte Adress	n+3	n+2	n+1	n
Little-Endian Byte Significance	MSByte			LSByte
Little-Endian Byte Order	n+3	n+2	n+1	n
Little-EndianByte-Reversed Order	n	n+1	n+2	n+3
Bit Label	0			31
Bit Significance	MSBit			LSBit

Tabela 2.2. Organizacija reči [3]

Big-Endian Byte Adress	Ν	n+1
Big-Endian Byte Significance	MSByte	LSByte
Big-Endian Byte Order	N	n+1
Big-Endian Byte-Reversed Order	n+1	n
Little-Endian Byte Adress	n+1	n
Little-Endian Byte Significance	MSByte	LSByte
Little-Endian Byte Order	n+1	Ν
Little-EndianByte-Reversed Order	N	n+1
Bit Label	0	15
Bit Significance	MSBit	LSBit

Tabela 2	2.3. Oı	ganiza	cija	pola	reči	[3]
----------	---------	--------	------	------	------	-----

Byte Address	Ν	
Bit Label	0	7
Bit Significance	MSBit	LSBit

#### Tabela 2.4. Organizacija bajta [3]

Kao što je već rečeno, sve *MicroBlaze* instrukcije su 32-bitne i razlikujemo dva tipa: tip A i tip B. Instrukcije tipa A imaju do dva izvorišna registra za operande i jedan odredišni registar za rezultat. Instrukcije tipa B imaju jedan izvorišni registar i jedan 16-bitni neposredni operand. Ove instrukcije imaju jedan odredišni registar. Prema funkciji koju obavljaju razlikujemo sledeće vrste instrukcija: aritmetičke, logičke, grananje, učitavanje/iščitavanje i posebne instrukcije. Sve instrukcije zahtevaju jedan takt, osim sledećih [3]:

- Učitavanje i iščitavanje podataka (dva takta)
- Množenje (dva takta)

#### • Grananje (tri takta)

*MicroBlaze* je *load/store* tip procesora što znači da može da učitava/iščitava podatke u/iz memorije. Ne može da obavlja nikakve operacije sa podacima u memoriji direktno; umesto toga, podatak iz memorije mora se učitati u *MicroBlaze* procesor i smestiti u registere opšte namene da bi se obavila bilo kakva operacija. Pristup podacima mora biti "poravnat" (tj. pristup rečima mora biti na granicama reči, pristup pola reči mora biti na pola reči), osim ako nije konfigurisan da podržava neporavnate izuzetke. Sav pristup instrukcijama mora biti poravnat na granicama reči. *Stack* konvencija korišćena u *MicroBlaze* procesorima počinje sa veće memorijske lokacije i ide dalje ka nižim memorijskim lokacijama. Podaci se iz memorije iščitavaju u suprotnom smeru; podatak sa najniže memorijske lokacije ide prvi itd. [4].

*MicroBlaze* procesor ima tridesetdva 32-bitna registra opšte namene i do osamnaest registara posebne namene, zavisno od konfiguracije. Registri opšte namene se obeležavaju od R0 do R31. Registri posebne namene su: *Program Counter* (PC) za određivanje adrese sledeće instrukcije, *Machine Status Register* (MSR) pokazuje status procesora npr. indikacija deljenja sa nulom, *Exception Address Register* (EAR) u koji se smešta *load/store* adresa koja je izazvala izuzetak, *Exception Status Register* (ESR) koji ukazuje kakav tip izuzetka se desio, *Floating Point Status Register* (FSR) koji ukazuje na stvari kao što su nevalidna operacija, deljenje sa nulom, *overflow*, *underflow* [4].

*MicroBlaze* takođe podržava *reset, interrupt, user exception, break* i *hardware* izuzetke. Za *interrupt* signale, *MicroBlaze* podržava samo eksterne *interrupt* izvore (koji se povezuju na *Interrupt* input port). Ukoliko su neophodni višestruki *interrupt* signali mora biti korišćen *interrupt* kontroler da upravlja višestrukim *interrupt* zahtevima ka *MicroBlaze* procesoru. *Interrupt* kontroler je dostupan za upotrebu u EDK softverskom alatu. Procesor će reagovati na *interrupt* signale ako je *Interrupt Enable* (IE) bit podešen na 1 u *Machine Status Register* (MSR). Takođe, dok traje prekidna rutina procesor onemogućava buduće *interrupt* signale brišući IE bit u MSR da bi se sprečilo gneždenje prekida što može izazvati nepredvidivosti u izvršavanju koda. IE bit je automatski postavljen opet kad se izvršava RTID instrukcija [4].

Postoji više softverskih alata koji se mogu koristiti za razvoj *embedded* sistema kao i više tipova čipova na kojima se oni mogu implementirati, od onih jeftinjih do skupljih koji nude brojne funkcionalnosti. Neke od njih su [3]:

- Spartan-3/3A/3AN/3A DSP/3E
- Spartan-6
- Virtex-4 FX/LX/SX
- Virtex-5 FX/LX/SX
- Virtex-6

*PowerPC* procesor je dostupan samo na pojedinim familijama (npr. *Viretx-4FX, Virtex-5FX* familije) jer on u suštini predstavlja fizički ugrađen procesor (dotični čipovi imaju hardverski ugrađen procesor). Naravno, i u familijama čipova koje imaju hardverski ugrađen procesor, može se i dalje implementirati i koristiti MicroBlaze *soft* procesor (koji će koristiti resurse opšte namene u čipu). Prednost *soft* pristupa *MicroBlaze* procesora je što se koristi onoliko hardverskih resursa koliko je potrebno. Na ovaj način korisnik može fleksibilno da izabere odgovarajući model koji je optimalan sa stanovištva njegovih potreba i potrošnje hardverskih resursa čipa.

#### 3.3. Performanse procesora

Osnovne performanse MicroBlaze procesora su [3]:

- Radna učestanost (maksimalna vrednost za sisteme sa malo zahtevnim periferijama):
  - 307 MHz na Virtex-6
  - 245 MHz na Virtex-5
  - 154 MHz na Spartan-6
  - 119 MHz na Spartan-3
- Potrošnja LUT elemenata čipa
  - 779/1134 LUT na Virtex-6
  - 240/330 LUT na Virtex-5
  - 770/1154 LUT na Spartan-6
  - 1258/1821 LUT na Spartan-3
- Vrste magistrala:
  - Master imaju mogućnost da iniciraju razmenu podataka
  - Slave mogu samo da odgovore na zahtev
  - Arbitraža se sastoji iz tri koraka
    - Uređaj koji želi da postane master šalje zahtev
    - Arbitar kontinuirano nadgleda zahteve i šalje individualne grant signale svakom master uređaju u skladu sa šemom master prioriteta i sa stanjem ostalih master zahteva u tom trenutku
    - Master koji je poslao zahtev čeka dok ne dobije pristupni grant. Kad postojeći master oslobodi magistralu, master tada koristi adresne i kontrolne magistrale da inicira razmenu podataka ka slave agentu.
  - Mehanizmi arbitraže:
    - Fiksni prioritet
    - > Round-robin
    - ➢ Hibridni

I *MicroBlaze* i *PowerPC* procesori koriste PLB v46 standard za komunikaciju sa svojom periferijom. Ovaj standard je dizajniran za komunikaciju sa periferijama koje su veoma zahtevne u pogledu brzine i propusnog opsega. Bitno je napomenuti da EDK koristi IP (*Intelectual Property*) jezgra koja su prilagođena ovom standardu. Takođe, sva IP jezgra koja su optimizovana za *MicroBlaze* procesore mogu se koristiti i za *PowerPC* i obratno.

*Xilinx embedded* sistem može biti praktično bilo koji računarski sistem. Tipično je projektovan da obavlja jednu funkciju tako da je mali i efikasan. Glavna prednost ovog tipa

procesora upravo je njegova fleksibilnost koja nam omogućava da obavlja različite funkcije. Na taj način značajno štedimo FPGA resurse i poboljšavamo performanse (pre svega brzinu) *embedded* sistema. Ovi sistemi su dizajnirani tako da budu što jeftiniji, minimalno troše energiju, da budu relativno brzi i da minimalno koriste FPGA resurse [3]. Važno je naglasiti i da se sistemi zasnovani na *embedded* procesorima lako modifikuju i ažuriraju (ako nema potrebe za izmenom periferije) pošto je potrebno izmeniti samo softverski kod po kom radi procesor, bez ikakve izmene u hardveru.

Softver koji upravlja sistemom je smešten u *block* RAM memoriji. Problem može biti što je na FPGA pločama ograničena veličina RAM memorije. Kada su ograničeni memorijskim kapacitetima, većina programera koristi *off-chip* memoriju, tako da je *Xilinx* razvio i nekoliko tipova memorijskih kontrolera koji obezbeđuju ovu funkcionalnost. Pored već pomenutog PLB standarda, *MicroBlaze* podržava i LMB standard koji se koristi za komunikaciju sa lokalnom *block* RAM memorijom. Takode, podržava i FSL standard koji se najčešće koristi za povezivanje sa posebnim DSP blokovima na kojima se vrše kompleksna izračunavanja i složenije matematičke operacije. *MicroBlaze* procesor ima do 16 FSL linkova. Na primer, za izračunavanje zbira dva broja bile bi potrebne 3 magistrale. Dve za slanje operanada ka DSP uređaju, i jedna za slanje rezultata ka procesoru. Upotreba FSL magistrala nam omogućava da "štedimo" nekoliko ciklusa na PLB magistrali, a takođe ne opterećujemo memorijske kapacitete [3,4].

## **4.** Embedded Development Kit

*Xilinx Embedded Development Kit* (EDK) je skup alata i *Intellectual Property* (IP) funkcionalnosti koji korisnicima omogućavaju da dizajniraju kompletan *embedded* sistem koji se može implementirati na FPGA čipu. Ovo poglavlje opisuje proces razvijanja ovakvog sistema koristeći EDK alat. U daljem tekstu su objašnjene mogućnosti koje su na raspologanju korisnicima tokom razvoja sistema. Data su precizna uputstva tako da čitaoci mogu odmah da krenu sa korišćenjem ovog alata.

*Embedded* sistemi su vrlo kompleksni. Sam razvoj hardverskog ili softverskog dela projekta je vrlo zahtevan poduhvat. Situacija se dodatno usložnjava kada treba ove dve komponente uskladiti da funkcionišu kao jedan sistem i konačno sve to implementirati na FPGA čipu. Kako bi pojednostavio ovaj proces, *Xilinx* nudi nekoliko setova alata, koji se jednim imenom zovu ISE *Design Suite*. Razlikujemo dve komponente EDK softvera: *Xilinx Platform Studio* (XPS) i *Software Development Kit* (SDK). XPS je razvojno okruženje koje se koristi za definisanje hardverskog dela sistema. XPS alatom se može upravljati koristeći *bash shell* komandnu liniju ili GUI, što će biti demonstrirano u ovom radu. SDK je integrisano razvojno okruženje komplemetarno sa XPS alatom koje služi za kreiranje i verifikaciju C/C++ aplikacija koje procesor izvršava [5].

Tipično, ISE razvojno okruženje se koristi za kreiranje *Embedded Processor source* fajla, koji je zatim dodat ISE projektu.

- U XPS alatu se vrši specifikacija mikroprocesora, periferija i interkonekcija između ovih komponenti.
- SDK se koristi za razvoj softvera.

#### 4.1. Base System Builder

BSB (*Base System Builder*) je vizard u XPS alatu koji brzo i efikasno formira radni dizajn. Nakon toga se ovaj dizajn može menjati i prilagođavati. *Xilinx* preporučuje korišćenje BSB vizarda za kreiranje osnove svakog novog *embedded* sistema. Takođe, u slučaju da se ukaže potreba za nekim promenama BSB značajno štedi vreme automatizovanjem standardnih zahteva za konfiguracijom hardverske i softverske platforme. U vizardu se formira radna verzija projekta koja sadrži sve osnovne elemente koji su neophodni za kompleksnije sisteme. BSB omogućava kreiranje *project* fajla; izbor ploče; izbor i konfiguraciju procesora i ulazno/izlaznih interfejsa; dodavanje periferija; kreiranje softvera; i generisanje izveštaja o projektu [5].

BSB prepoznaje komponente sistema i konfiguraciju izabrane ploče, i omogućava adekvatne opcije u skladu sa potrebama korisnika. Kao što je već rečeno, BSB pruža više mogućnosti za izbor odgovarajuće ploče. Može se izabrati neka od opcija koje su na rasploganju, od *Xilinx* proizvoda ili od njegovih partnera. Nakon što se izaberu i odgovarajuće periferije, BSB kreira UCF (*User Constraints File*) fajl koji dodeljuje pinove izabranim periferijama. Takođe, BSB kreira i kompletnu platformu i test aplikacije koje su spremne da budu spuštene i pokrenute na ploči. Za svaku odabranu opciju postoje podrazumevane vrednosti, koje se mogu menjati u XPS alatu. Ukoliko se dizajn vrši za određenu ploču, BSB omogućava izbor vrste procesora (*MicroBlaze* ili *PowerPC*,

zavisno od izabranog FPGA čipa pri čemu je *PowerPC* ponuđen samo za one čipove koji imaju hardverski ugrađen procesor), sa različitim kompatibilnim i često korišćenim periferijama. Tako dobijamo početnu hardversku platformu. Naknadno se može dodati još procesora ili periferija, ako je potrebno. Mogu se podešavati sledeće opcije kod procesora [5]:

- Frekvencija referentnog takta
- Frekvencija takta na procesorskoj magistrali
- Konfiguracija procesora za debagovanje
- Podešavanje keša
- Floating Point Unit (FPU) podešavanja

BSB razume koji tipovi eksterne memorije i ulazno/izlaznih uređaja su dostupni na izabranoj ploči i omogućava korisnicima da izabere sledeće:

- Tip uređaja
- Tip perfiferije
- Brzina u baudima
- Parity
- Korišćenje signala prekida (*interrupts*)

Za eksternu memoriju i ulazno/izlazne uređaje je dostupna i detaljna dokumentacija. BSB vizard pruža mogućnost da se ubace dodatne periferije. Periferije moraju biti podržane od strane izabrane FPGA ploče. Nakon što se odaberu ulazni i izlazni uređaji u BSB vizardu, generišu su i jednostavne C aplikacije. Za razvoj softvera je preporučen SDK alat, o čemu će više reči biti kasnije. Nakon što se izaberu odgovarajuća podešavanja, BSB prikazuje sumarni izveštaj o sistemu. Pre nego što se završi sa generisanjem projekta korisnik se može vratiti na neku od prethodnih stranica i promeniti neka podešavanja. U ovom radu biće korišćen *Virtex-ML507* ploča sa *MicroBlaze* procesorom. Neke ploče nude i mogućnost izbora *PowerPC* procesora. U gotovo svim slučajevima način korišćenja alata je isti kao i za *MicroBlaze*. Napomenimo da se konfiguracija procesora može vršiti i za ploču koja nije ponuđena (*proprietary* ploča), ali tada se pojedina podešavanja moraju dodatno uraditi pošto alat nema podatke o takvoj ploči u svojoj bazi (npr. moraju se podesiti lokacije pinova, definisati periferije koje su na raspolaganju i dr.).

### 4.2. Kreiranje novog embedded projekta

U ovom potpoglavlju, sledi detaljan opis razvoja *embedded* sistema, korak po korak. Kreiraće se novi projekat, u koji je uključen *embedded* procesor. ISE automatski startuje XPS alat i otvara BSB kako bi kompletirali dizajn. XMP fajl je *top-level* fajl koji sadrži opis *embedded* sistema. Sve informacije o projektu su sačuvane u ovom fajlu. XMP je kreiran i njime se upravlja u ISE alatu, kao i sa drugim *source* fajlovima, kao što su HDL kod ili UCF fajlovi [5].

Dakle, prvi korak je pokretanje ISE *Project Navigator* alata. Ukoliko je neki projekat otvoren, trebalo bi ga zatvoriti, klikom na opciju *File > Close project*, slika 4.2.1. Zatim se formira novi projekat sa *embedded* processor sistemom kao *top level* entitetom, izborom opcije *File > New Project*, slika 4.2.2. Prilikom kreiranja projekta, biraju se opcije nabrojane u tabeli 4.2.3 u skladu sa činjenicom da je projekat namenjen za rad sa pločom *ML507*.

ISE	ISE	Project N	Navigato	r (M.63c)	- C:\Users\	Srdjan\[	Deskto	op\KOM	PL\p	prove	ra\prove	ra.xise - [Des	ign Su	mma	ry]
E	File	Edit	View	Project	Source	Process	s To	ols W	inde	ow	Layout	Help			
		New Pro	oject			50	a   .	» 🏓 🍠	P	8	BP	2 🔊	G 8		📼 🎤 K?
D		Open P	roject			P X		Des	ign	Over	view			-	
		Open Ex	ample			La Kana				Sum	mary				
4		Project	Browser.			Jiation	0			IOB	Propertie	s			Project File:
d		Conv Pr	niect							Mod	lule Leve	Utilization			Module Name
		Class	ojectin							Timi	ing Const	traints		=	Target Device
E C		Close Pl	roject				(5			Pino	ut Repor	t		-	Target Device
		New			Ctrl+N				貝	Cloc	k Report				Product Versi
1	-	TAC.VV			Carrie		HINE		9	Stati	c liming				Design Goal:
9	ø	Open			Ctrl+O		28	B. XPS	Err	ors ar	nd Warnı	ngs			Decian Strate
		Close					20			Plate	gen Mess	ages		1	Design Strate
E	-						M			Libg	en Messa	ages		1.1	Environment:
		Save			Ctrl+S					Simo	gen Mess	ages		1	



ISE I	SE Proje	ct Navig	ator (M.63	c)	_		_	_		 _		-
File	Edit	View	Project	Source	Process	Tools	Window	Layout	Help			
	New Pr	oject			100	(a) >>>	P P	BB	۶ 🔊		<b>B</b> .	<u>ب</u> الج
	Open P	roject				c l						
	Open E	xample				-						
	Project	Browser			uite							
	Сору Р	roject			L							
	Close P	roject										
	New			Ctrl+N								
2	Open			Ctrl+O	L							
	Close					;						
	Save			Ctrl+S	open							

Slika 4.2.2. Otvaranje novog projekta

Wizard Screen	System Property	Setting or Command to Use
	Name	Odabrati ime projekta (bez razmaka). U ovom primeru, ime je <i>provera</i> .
Create New Project	Loacation and Working Directory	Odabrati lokaciju i radni direktorijum (bez razmaka). U ovom primeru putanja je <i>Desktop/Master</i> .
	Description	Može se dodati i opis projekta (opciono).
	Top-level source type	Select HDL (podrazumevano)
	Product Category	All
Project Settings	Family	Virtex5
	Device	XC5VFX70T
	Package	FF1136
	Speed	-1
	Synthesis Tool	XST (VHDL/Verilog)

	Simulator	User-specific
	Preferred Language	VHDL
	Ostalo prihvatiti podrazumevan	podešavanja
Project Summary	Pokazuje pregled izabranih opcija u New Project vizardu.	Bez promena.

#### Tabela 4.2.3. Podešavanja koja treba odabrati prilikom kreiranja novog projekta

Kad se završi sa definisanjem projekta, klikne se na dugme *Finish, New Project* vizard se zatvara i ISE *Project Navigator* otvara projekat koji smo upravo kreirali. Sledeći korak je dodavanje *source* fajla, korišćenjem *New Source* vizarda. U opcijama se klikne na *Project > New Source* dugme kao na slici 4.2.4, i biraju se opcije navedene u tabeli 4.2.5.

ISE IS	E Project Navig	gator (M.63c) - C:\Users\Srdjan\Desktop\Master\provera\provera.xise	
File	Edit View	Project Source Process Tools Window Layout Help	
	2 🗟 🖉	📑 New Source 🖻 🔎 🙊 🙊 🏓 💽 😤 🖻	
Desig	IN	Add Source	
*	View: 💿 🔯 I	I 🚰 Add Copy of Source	
6	Hierarchy	New VHDL Library	
8	🗄 🝯 prover	Manual Compile Order	
	E XCSVfx	Import Custom Compile File List	
		Disable Hierarchy Reparsing	
		Force Hierarchy Reparse	
		Cleanup Project Files	

Slika 4.2.4. Dodavanje novog Source fajla u projekat

Wizard Screen	System Property	Setting or Command to Use
Select Source Type	Source Type	Embedded Processor
	File name	system
	Location	Prihvatiti podrazumevanu
		lokaciju.
	Add to project	Ostaviti čekirano
Project Summary	Pokazuje pregled izabranih opcija u New Source vizardu	Bez promena

#### Tabela 4.2.5. Podešavanja koja treba odabrati prilikom dodavanja novog Source fajla

Nakon završetka, ISE prepoznaje da smo izabrali *embedded* sistem i pokreće XPS. Trebalo bi da se pojavi dijalog prozor, sa pitanjem da li se želi kreiranje *Base system* projekta upotrebom BSB vizarda. Nakon prihvatanja, kreira se projekat sa podešavanjima datim u tabeli 4.2.6. (ukoliko su neka podešavanja preskočena u tabeli, za ta podešavanja su korišćene podrazumevane (difolt) vrednosti.

Wizard Screen	System Property	Setting or Command to Use
Welocome to the Base System Builder	Project type options	I would like to create a new design
Board selection	Board Vendor	Xilinx
	Board Name	Virtex 5 ML507 Evaluation Platform
	Board Revision	A
System Configuration	Type of system	Single-Processor System
Processor Configuration	Processor Type	MicroBlaze
	System Clock Frequency	100 MHz
	Local Memory	64 KB
	Enable Floating Point Unit	Otčekirati ovu opciju
Peripheral Configuration	Processor 1 (MicroBlaze) Peripheral list	Ukloniti sledeće periferije sa liste podrazumevanih vrednosti "Processor 1 (MicroBlaze)": IIC_EEPROM SRAM DDR2_SRAM Ethernet_MAC Hard_Ethernet_MAC Hard_Ethernet_MAC LEDs_Positions SysACE_CompactFlash FLASH RS232_Uart_1 RS232_Uart_2 PCIe Bridge
Cache Configuration		Kliknuti Next
Application Configuration	Example Application Options	Ostaviti podrazumevane vrednosti
Summary	System Summary Page	

#### Tabela 4.2.6. Podešavanja koja treba odabrati u BSB alatu

Napravimo pregled odabranih opcija. Prvo je ponuđeno da li da se kreira novi dizajn ili da se učita postojeći. Pošto ništa još nije kreirano, bira se kreiranje novog dizajna. Dalje se nudi izbor FPGA ploče. U ovom primeru koristiće se *Xilinx Virtex 5 ML507* ploča. BSB nudi mogućnost razvijanja dizajna i sa dva procesora, ali će se ovde koristiti sistem sa jednim procesorom (*single processor*). Izabran je *MicroBlaze* procesor, sa radnom učestanošću od 100 MHz i lokalnom

memorijom od 64 KB. Ovde je izabrana maksimalna vrednost RAM memorije, mada je i ona dosta skromna. Ovo može prestavljati problem, jer aplikacija koje se izvršava na procesoru mora biti optimizovana da stane u dodeljenu memoriju. Na to treba obratiti pažnju prilikom pisanja C koda. Od periferija, u ovom primeru će biti korišćene samo LED diode, DIP svičevi i *Push* dugmad. Ostale ponuđene periferije treba ukloniti, selektovanjem date periferije i klikom na *Remove*. Ostalo treba prihvatiti podrazumevane (difolt) vrednosti. Kad se završi sa svim selekcijama, klikne se na *Finish*, a onda u dijalog prozoru *OK*.

## 4.3. Xilinx Platform Studio

Nakon što je kreirana osnova projekta uz pomoć BSB vizarda, XPS nudi mogućnost ažuriranja i nadogradnje projekta. Na slici 4.3.1. je dat izgled glavnog prozora u XPS. Ovaj prozor je podeljen na tri dela:

- Project Information Area (1)
- System Assembly View (2)
- Console Window (3)



Slika 4.3.1. Izgled XPS prozora[5]

U glavnom prozoru postoje i tri labele koje identifikuju sledeće obalsti:

- *Connectivity Panel* (4)
- View Buttons (5)
- *Filters* (6)

#### 4.3.1. Project Information Area

*Project Information Area* nudi informacije o projektu i omogućava njegovu kontrolu. Ovaj deo prozora sadrži: *Project, Applications* i *IP Catalog* tabove.

i) Project Tab

*Project* tab, prikazan na slici 4.3.1.1 a), daje listu referenci na fajlove vezane za projekat. Postoje tri kategorije informacija[5]:

- Project Files sadrži: Microprocessor Hardware Specification (MHS) fajl, Microprocessor Software Specification (MSS) fajl, User Constarints File (UCF), iMPACT Command fajl, Implentation Option fajl i Bitgen Option fajl
- Project Options pruža informacije kao što su: Device, Netlist, Implementation, Hardware Description Language (HDL) i Sim Model
- *Design Summary* sadrži grafički izgled stanja *embedded* dizajna i daje lak pristup sistemskim fajlovima

Project	↔□₽×	Applications	↔ [	3 & X	IP Catalog		⇔□₽×
Platform		Software Projects	5	1	<mark>:</mark> ⊕		
<ul> <li>Project Files</li> <li>MHS File: system.mhs</li> <li>MSS File: system.mss</li> <li>UCF File: data/system.uc</li> <li>iMPACT Command File:</li> <li>Implementation Options</li> <li>Bitgen Options File: etc/</li> <li>Project Options</li> <li>Device: xc5vfx70tff1136-1</li> <li>Netlist: SubModule</li> <li>Implementation: Project</li> <li>HDL: vhdl</li> <li>Sim Model: BEHAVIORAL</li> <li>Design Summary</li> </ul>	:f etc/download. ; File: etc/fast_ru bitgen.ut I L Navigator	Add Soft Default: r Default: r Project: Processor Executabl Compiler Sources Processor Executabl Processor Executabl Processor Executabl Compiler Sources Headers Headers	ware Application Project microblaze_0_bootloop microblaze_0_xmdstub TestApp_Memory_micr :: microblaze_0 e: C:\Users\Srdjan\Deskt Options TestApp_Peripheral_mid :: microblaze_0 e: C:\Users\Srdjan\Deskt Options	t roblaze top\Ma croblaz top\Ma	Description Example 2 EDK Ins Control Analog Arithme Control Arithme Control Arithme Con	IP Version	IP Type
a)		]	b)			c)	

Slika 4.3.1.1. a) Project Tab, b) Applications Tab, c) IP Catalog Tab

ii) Applications Tab

*Applications* tab, prikazan na slici 4.3.1.1 b), daje listu podešavanja opcija softvera, *header* fajlova, i *source* fajlova koji su povezani sa aplikacijom. U ovom tabu može se [5]:

- Kreirati i dodati softver, izgraditi projekat, i smestiti u RAM memoriju
- Podesiti opcije kompajlera

• Dodati source i header fajlovi u projekat

Bitno je napomenuti da, iako se softver može kreirati i njime upravljati u XPS, ipak se za te svrhe preporučuje korišćenje SDK alata.

iii) IP Catalog Tab

IP Catalog tab, prikazan na slici 4.3.1.1 c), pruža informacije o IP jezgrima, uključujući[5]:

- Ime jezgra i status licence (nelicenciran, zaključan, ili otključan)
- Verzija i status
- Podržani procesori
- Klasifikacija

Dodatne informacije o IP jezgrima, uključujući istoriju izmena, specifikaciju i *Microprocessor Peripheral Description* (MPD) fajl, su dostupne kad se klikne desno dugme na mišu na IP jezgru u *IP Catalog* tabu. IP jezgra su grupisana hijerarhijski po funkciji.

#### 4.3.2. System Assembly View

System Assembly View pruža mogućnost da se vide i konfigurišu elementi sistema. Ako prethodno nije maksimizovan, otvoriti System Assembly View tab na dnu panela.

i) Bus Interface, Ports, and Addresses Tab

System Assembly View sadrži tri panela, koji se mogu otvoriti klikom na odgovarajuću opciju:

- *Bus Interface* tab prikazuje magistrale korišćene u dizajnu. Koristi se za izmenu informacija i konekcija za svaku magistralu
- *Ports* tab prikazuje portove korišćene u u dizajnu. Sadrži detalje i opcije za promenu portova
- *Addresses* tab prikazuje opseg adresa za svaku IP instancu u projektu. Klikom na opciju *Generate Addresses* automatski se generiše adresna mapa sistema.
- *ii)* Connectivity Panel

Kada se izabere *Bus Interface* tab, vidi se *Connectivity Panel*, koji predstavlja grafičku reprezentaciju konekcija hardverske platforme. Prelaskom mišem preko *Connectivity Panel* vide se dostupne konekcije na magistralama. Vertikalna linija predstavlja magistralu, a horizontalna linija predstavlja interfejs magistrale ka IP jezgru. Ako je moguće napraviti kompatibilnu konekciju, konektor je prikazan na preseku magistrale i interfejsa ka IP jezgru. Linije su u boji, kako bi pokazale kompatibilnost magistrala. Različiti simboli pokazuju da li je IP blok povezan na magistralu kao *master* (kvadrat) ili *slave* (krug). Prazni konektor pokazuje konekciju koja se može uspostaviti. Popunjeni konektor predstavlja postojeću konekciju. Da bi se konekcija uspostavila ili raskinula klikne se na simbol konektora.

iii) Filters

XPS omogućava filetere koji se mogu korisititi da se promeni izgled Bus Interface i Ports taba u System Assembly View. Filteri su izlistani u Filters, kada su Bus Interface ili Ports tabovi izabrani. Korišćenje ovih filtara može uprostiti izgled *connectivity panel* kad se kreira dizajn sa velikim brojem različitih magistrala.

#### iv) View Buutons

Kako bi se mogle lakše sortirati informacije i revidirati dizajn, *System Assembly View* omogućava dva dugmeta koji menjaju način na koji su podaci prikazani:

- Change to Hierarchial/Flat View dugme
  - Podrazumevani izgled se zove hijerahijski izgled. Informacije koje su prikazane za dati dizajn su bazirane na instancama IP jezgara na hardverskoj platformi i organizovane u obliku proširivog stabla.
  - U flat view, informacije su sortirane alfanumerički po svakoj koloni
- Expand/Collapse All Tree Nodes dugme

Ikonica +/- proširuje ili skuplja sve veze između magistrale i IP jezgara i omogućava brzo i lako povezivanje periferija sa magistralom.

#### v) Console Window

*Console window* omogućava povratne informacije tokom izvršavanja nekih opcija. Postoje tri taba: *Console, Warnings* i *Errors*.

vi) Start Up strana

*Start Up* strana sadrži relevantne informacije vezane za verziju XPS koja se koristi, uključujući set linkova sa različitim informacijama. Takođe, tu je i tab koji olakšava lociranje EDK dokumentacije. Ukoliko *Start Up* strana nije već otvorena, može se otvoriti klikom na *Help* > *View Start Up Page*.

#### vii) Directory View

BSB automatski kreira direktorijum projekta sa imenom *embedded* sistema koji je kreiran. U ovom direktorijum su sadržani poddirektorijumi vezani za projekat [5]:

 $\_xps$ 

Sadrži fajlove generisane od strane XPS i drugih alata za interni menadžment projekta. Ovaj direktorijum korisnici ne koriste.

• blockdiagram

Sadrži fajlove vezane za blok dijagram

• data

Sadrži UCF fajl.

• etc

Sadrži fajlove koji sadrže opcije koje služe za korišćenje raznih alata. Ovaj direktorijum je prazan ako se ne primenjuju nikakve akcije van BSB.

• pcores

Koristi se za dodavanje periferija. Detaljnije će biti objašnjen u nastavku.

Druga dva direktorijuma sadrže fajlove generisane od strane BSB:

• *TestApp\_Memory\_microblaze\_0* 

#### • TestApp\_Peripheral\_microblaze\_1

Ovi direktorijumi sadrže C kod test aplikacije i *header* fajlove. Iako su dostupni za upotrebu, u ovom radu će se koristiti aplikacija koja će biti razvijena u SDK alaltu, o čemu će biti reči kasnije.

U direktorijumu glavnog projekta prikazanom na slici 4.2.3.1, postoji takođe nekoliko fajlova. Neki od njih su [5]:

• system.xmp

Ovo je *top-level* fajl. XPS čita ovaj fajl i grafički prikazuje njegov sadržaj u XPS korisničkom interfejsu.

• system.mhs

MHS fajl sadrži elemente sistema, njihove parametre, i konekcije u tekstualnom formatu. MHS fajl je hardverska osnova projekta.

• system.mss

MSS fajl sadrži softverski deo dizajna, opisuje elemente sistema i različite softverske parametre vezane za periferije u tekstualnom formatu. MSS fajl je softverska osnova projekta.

MHS i MSS fajlovi su glavni proizvod razvijanog XPS dizajna. Kompletan hardver i softver sistema je sadržan u ovim fajlovima.

A Naster	provera > system >			
Organize 🔻 Include	in library ▼ Share with ▼ Burn	New folder		
🔆 Favorites	Name	Date modified	Туре	Size
🧫 Desktop	🔐xps	16/11/2014 18:35	File folder	
\rm Downloads	퉬 blockdiagram	16/11/2014 12:46	File folder	
🔚 Recent Places	퉬 data	16/11/2014 12:46	File folder	
	퉬 etc	16/11/2014 12:46	File folder	
🥽 Libraries	퉬 hdl	16/11/2014 18:36	File folder	
Documents	implementation	16/11/2014 18:39	File folder	
👌 Music	imicroblaze_0	16/11/2014 18:25	File folder	
Pictures	🌗 pcores	16/11/2014 12:37	File folder	
📑 Videos	퉬 synthesis	16/11/2014 18:39	File folder	
	TestApp_Memory_microblaze_0	16/11/2014 12:46	File folder	
🖳 Computer	TestApp_Peripheral_microblaze_0	16/11/2014 12:46	File folder	
🏭 Local Disk (C:)	clock_generator_0	16/11/2014 18:36	Text Document	1 KB
	📄 platgen	16/11/2014 18:38	Text Document	12 KB
📬 Network	Dlatgen.opt	16/11/2014 18:38	OPT File	1 KB
	system.bsb	16/11/2014 12:46	BSB File	4 KB
	system.create.tcl	16/11/2014 12:37	TCL File	2 KB
	📄 system	16/11/2014 18:39	Text Document	2 KB
	system.make	16/11/2014 18:35	MAKE File	11 KB
	system.mhs	16/11/2014 12:46	MHS File	5 KB
	system.mss	16/11/2014 12:46	MSS File	2 KB
	system.setprop.tcl	16/11/2014 18:35	TCL File	1 KB
		16/11/2014 18:35	Xilinx Platform Stu	2 KB
	system_incl.make	16/11/2014 18:35	MAKE File	10 KB
	XpsGuiSessionLock	16/11/2014 18:25	File	0 KB

Slika 4.3.2.1. Pregled direktorijuma projekta

## 4.4. Pregled hardverske platforme

Hardverska platforma *embedded* sistema uključuje jedan ili više procesora, zajedno sa različitim brojem periferija i memorijskih blokova. Ovi IP blokovi koriste *interconnect* mrežu za komunikaciju. Portovi se povezuju na "spoljni svet", što može biti ostatak FPGA platforme, ili van FPGA platforme uopšte. Ponašanje svakog procesora ili periferije može biti podešeno. Od parametara implementacije zavise opcione karakteristike i specifikacija kako je hardverska platforma implementirana na FPGA ploči.

#### 4.4.1. Razvoj hardverske platforme u Xilinx Platform Studio alatu

XPS obezbeđuje interaktivno razvojno okruženje koje omogućava da se specificiraju svi aspekti hardverske platforme. Kao što je već rečeno, opis hardvera je sadržan u MHS fajlu. MHS fajl, koji je tekstualni fajl koji se može menjati, je *source* fajl koji predstavlja hardversku komponentu *embedded* sistema. Tu su sadržane sve instance periferija zajedno sa njihovim parametrima, kao i konfiguracija *embedded* sistema uključujući arhitekturu magistrala, periferija, procesora, konekcija i adresnog prostora. Sadržaju MHS fajla se može prstupiti na sledeći način:

- Izabere se *Project* tab u *Project Information Area* u XPS
- U *Project Files* se dva puta klikne na MHS File: system.mhs
- Primera radi, izborom opcije *Edit* > *Find*, pretražiti *xps\_uart16550*. U fajlu se nalazi više ovih instanci. Ukoliko se nalazi u okviru *Begin/End* para, onda se radi o portu koji je deo IP. Ukoliko se *PORT* nalazi pri vrhu MHS fajla, onda povezuje *embedded* sistem sa okruženjem.

#### 4.4.2. Hardverska platforma u System Assembly View

System Assembly View u XPS alatu prikazuje IP instance hardverske platforme u obliku proširivog stabla ili u obliku tabele. IP elementi, njihovi portovi, podešavanja i parametri se mogu konfigurisati u System Assembly View i sve promene se zapisuju direktno u MHS fajl. Izmena imena porta ili nekog parametra se izvršava nakon pritiska dugmeta *Enter* ili klikom na OK. XPS sve modifikacije sistema automatski zapisuje u MHS fajl. Direktno menjanje MHS fajla nije preporučljivo, naročito ne za početnike.

#### *i)* Eksportovanje hardverske platforme

Kako je već rečeno, u EDK alatu hardverski deo sistema se razvija u XPS alatu, dok se softver razvija i debaguje u SDK alatu. SDK zahteva informacije u vezi sa hardverskom platformom, pa je neophodno eksportovati system.xml fajl. U ovom fajlu su informacije koje SDK zahteva kako bi bilo moguće razvijati softver na hardverskoj platformi koja je dizajnirana. To se vrši na sledeći način:

- U XPS alatu se izabere *Project* > *Export Hardware Design to SDK*
- Lokacija direktorijum je već izabrana i ne može biti promenjena kad se dizajn eksportuje iz XPS. Podrazumevana putanja je system/SDK/SDK\_Export/hw u direktorijumu projekta.
- Izabere se opcija *Export Only*.

Vrlo je važno razumeti šta se dešava prilikom operacije eksportovanja. Kad se izabere opcija *Export Only*, program kreira nekoliko fajlova koji su korišćeni od strane SDK alata. Pored XML fajla, uključena je i dokumentacija softverskih drajvera i IP hardvera, tako da se njima može

pristupiti i iz SDK. Druga opcija, *Export&Launch SDK*, eksportuje dizajn i automatski otvara SDK gde se dalje razvija softverski deo projekta.

#### ii) Generisanje bitstream fajla

Nakon opisa hardvera *embedded* sistema u XPS alatu, koristi se ISE *Project Navigator* za implementaciju dizajna i generisanje *bitstream* fajla. C kod nije deo ovog bitstream fajla i on će biti dodat kasnije u SDK. Postupak je sledeći.

- Otvori se ISE *Project Navigator*
- U Project Navigator softveru se izabere *Project > Add Source* i izabere se *system/data/system.ucf* fajl. Ovaj korak je neophodan, jer *ISE Place and Route* alat zahteva informacije kao, na primer, na koje pinove se vezuju odgovarajuće periferije i slično.
- U dijalog prozoru se klikne OK
- Klikne se na system.xmp u Design prozoru
- U *Process: system* se klikne dva puta na *Generate Programming File*, kako bi se kreirao *bitstream* fajl. Za ovaj proces je potrebno nekoliko minuta, dok se ne pojavi poruka "*Process "Generate Programming File" completed successfully*".
- Generisani *bitstream* fajl se zove system.bit. Pored njega, tu je još i fajl edkBmmFile\_bd.bmm, koji služi da SDK učita sadržaj instrukcijske (ROM) memorije na željenu ploču. Ovi fajlovi će biti korišćeni prilikom razvoja aplikacije.

![](_page_22_Picture_9.jpeg)

Slika 4.4.2.1. System.ucf fajl dodat u projekat

ISE Project Navigator (M.63c) - C:\Users\Srdjan\E	Desktop\Master\provera\provera.xise - [Design Summar
E File Edit View Project Source Process	; Tools Window Layout Help
o or 🗙 🗊 🖉 🖉 📲 🖗 🔲	» アアタタア 🗟 🔽 着 🗆 🗆
Design       Implementation         View:       Implementation         Implementation       Implementation <th><ul> <li>Design Overview <ul> <li>Design Overview</li> <li>Summary</li> <li>IOB Properties</li> <li>Module Level Utilization</li> <li>Timing Constraints</li> <li>Pinout Report</li> <li>Clock Report</li> <li>Static Timing</li> <li>Clock Report</li> <li>Static Timings</li> <li>Errors and Warnings</li> <li>Parser Messages</li> <li>Synthesis Messages</li> <li>Synthesis Messages</li> <li>Translation Messages</li> <li>Map Messages</li> <li>Place and Route Messages</li> </ul> </li> </ul></th>	<ul> <li>Design Overview <ul> <li>Design Overview</li> <li>Summary</li> <li>IOB Properties</li> <li>Module Level Utilization</li> <li>Timing Constraints</li> <li>Pinout Report</li> <li>Clock Report</li> <li>Static Timing</li> <li>Clock Report</li> <li>Static Timings</li> <li>Errors and Warnings</li> <li>Parser Messages</li> <li>Synthesis Messages</li> <li>Synthesis Messages</li> <li>Translation Messages</li> <li>Map Messages</li> <li>Place and Route Messages</li> </ul> </li> </ul>
Running: Platform Generator      Processes: system	Bitgen Messages
<ul> <li>Design Summary/Reports</li> <li>Design Utilities</li> <li>User Constraints</li> <li>Synthesize - XST</li> <li>Implement Design</li> <li>Generate Programming File</li> <li>Configure Target Device</li> <li>Update Bitstream with Processo</li> <li>Export Hardware Design To SDK</li> <li>Analyze Design Using ChipScope</li> </ul>	Synthesis Report         Design Properties         Enable Message Filtering         Optional Design Summary Contents         Show Clock Report         Show Failing Constraints         Show Warnings         Show Errors

Slika 4.4.2.2. Generisanje bitstream fajla

SDK omogućava razvoj aplikacije koja se izvršava u *embedded* sistemu. SDK je komplementarna sa XPS. Dizajn počinje kreiranjem softverskog projekta. SDK okruženje može upravljati sa više softverskih projekata. Kada se softverski projekat kreira, SDK zahteva da se kreira hardverska platforma i *board support package*. Hardverska platforma je dizajn *embedded* hardvera koji je kreiran u XPS. Ona uključuje XML, *bitstream* i BMM fajl. Kad se u SDK importuje XML fajl, tada ustvari importujemo hardversku platformu. U ISE *Design Suite 12* alatu, moguće je da više hardverskih platformi postoji u jednom radnom okruženju. O radu u SDK softveru, više reči će biti u narednom poglavlju, gde će biti objašnjeno kako se implementira željena softverska funkcionalnost u *embedded* sistemima.

## 5. Implementacija provere i ažuriranja IPv4 zaglavlja

*Embedded* sistemi se najčešće koriste za obavljanje nekih specifičnih zadataka. Prednost ovakve realzacije je što zauzimaju relativno malo resursa, imaju pristojne performanse i pružaju veliku fleksibilnost u smislu dodavanja novih i menjanja postojećih funkcionalnosti. U ovom radu je realizovana provera i ažuriranje IPv4 zaglavlja. Kao što je već pomenuto, ovo je funkcija koja se izvršava u ruterima i na osnovu koje utvrđujemo da li je došlo do greške prilikom prenosa paketa. Ova funkcionalnost je implementirana u posebnoj periferiji, koja je naknadno razvijena i dodata u postojeći projekat. Od ulazno/izlaznih periferija sadrži DIP svičeve i LED diode. Osim toga, ima i 12 registara koje služe za skladištenje odgovarajućih informacija. Ovde je dat pojednostavljen primer, čiji je cilj da se objasne mogućnosti koje EDK alat pruža, i način korišćenja istog. Izborom odgovarajuće kombinacije na DIP sviču, korisnik bira određeno zaglavlje. C aplikacija koje je razvijena u SDK, obrađuje to zaglavlje i pali odgovarajuću diodu zavisno da li je validno ili nije. Takođe, ukoliko je zaglavlje validno ono se ažurira i smešta u registre u ovoj periferiji. Na ploči ima 8 LED dioda, tako da 4 služe za brojanje korektnih zaglavlja, a 4 za brojanje pogrešnih.

#### 5.1. Create and Import Peripheral vizard

XPS značajno olakšava dizajn *embedded* sistema jer automatizuje dosta operacija, i svodi ih na seriju odgovarajućih selekcija. Prilagođavanje dizajna može biti jednostavno poput podešavanja par parametara na nekom od IP jezgara ili kompleksnije poput razvoja potpuno novog IP jezgra i integrisanja u postojeći dizajn. Za to se koristi CIP vizard koji pruža slične mogućnosti kao i BSB. On obezbeđuje radni okvir za dizajn, uključujući logiku za interfejse magistrale kao i HDL templejt fajl tako da je znatno olakšano integrisanje logike koju korisnik razvija. Svi fajlovi koji su neophodni za uključivanje korisničkog IP jezgra (*pcore*) u *embedded* sistem su obezbeđeni od strane CIP vizarda. U ovom poglavlju je objašnjeno šta se dešava tokom razvoja korisnički definisanog IP jezgra. Postupak je sledeći:

• Pokrene se ISE *Project Navigator*, otvori se prethodni projekat, klikne na system.xmp, i dvostruki klik na *Manage Processor Design* (locirano u okviru *Design Utilities*) slika 5.1.1, kako bi se pristupilo XPS alatu.

![](_page_24_Picture_5.jpeg)

Slika 5.1.1. Pokretanje XPS alata iz ISE softvera

Xilinx Platform Studio - C:\Use	rs\Srdjan\Desktop	\Master\provera\syst	em\system.xmp - [S	ystem Assembly \	View]	
🔶 File Edit View Project	Hardware Soft	ware Device Config	uration Debug	Simulation Wir	ndow Help	
i 🗋 ờ 🗟 🖗 🕼 は ն	Btg Generate Net	list	M 🛛 🗗 🔂	🔽 🔀 🧆 🗄	Bas 🔝 🎇 🛛 🗤	ÞG 📥 🐚
Project	Generate Bits	tream	Bus Interfaces	Ports Addr	esses	
Platform	🌺 Create or Imp	ort Peripheral	Name	Bus Name	IP Туре	IP Version
Project Files	Configure Co	processor	- dlmb		🚖 lmb_v10	1.00.a
MHS File: system.mhs	in configure co		ilmb		📩 lmb_v10	1.00.a
MSS File: system.mss	Launch Clock	Wizard	mb_plb		📩 plb_v46	1.04.a
UCF File: data/system.ucf	Check and Vi	ew Core Licenses	🗄 microblaze_0		📩 microblaze	7.30.b
iMPACT Command File: e			🗄 lmb_bram		🙀 bram_block	1.00.a
Implementation Options	👺 Clean Netlist		🗄 dlmb_cntlr		📩 Imb_bram_i	2.10.b
Bitgen Options File: etc/bi	🥦 Clean Bits		🗄 ilmb_cntlr		📩 Imb_bram_i	2.10.b
Project Options			🗄 mdm_0		📩 mdm	1.00.g
Device: xc5vfx70tff1136-1	B Clean Hardw	are	proc_sys_re		🛉 proc_sys_re	2.00.a
Metlist: TopLevel		<u> </u>	DIP_Switche		🙀 xps_gpio	2.00.a
Implementation: Project N	lavigator	<u> </u>	LEDs_8Bit		🐈 xps_gpio	2.00.a
HDL: vhdl		3	• Push_Butto		🙀 xps_gpio	2.00.a
Sim Model: BEHAVIORAL		20 <del>0</del> 02	clock_gener		🚖 clock_gene	4.00.a
Design Summary	1.1.1					

• U XPS alatu se otvori *Hardware > Create or Import Peripheral*, slika 5.1.2.

Slika 5.1.2. Opcija za kreiranje ili importovanje periferije

• Otvoriće se *Peripheral Flow* stranica, slika 5.1.3. Ovde se može izabrati da li će se kreirati nova ili importovati postojeća periferija. Odabere se opcija *Create templates for a new peripheral*.

Create and Import Peripheral Wizard	And a second sec	? ×
Peripheral Flow Indicate if you want to create a new periphera	or import an existing peripheral.	
This tool will help you create templates for a new E project or EDK repository. The interface files and d	DK CoreConnect peripheral, or help you import an existing EDK CoreConnect peripheral in irectory structures required by EDK will be generated.	to an XPS
	Select flow	
	<ul> <li>Create templates for a new peripheral</li> <li>Import existing peripheral</li> </ul>	
	Flow description	
1	This tool will create HDL templates that have the EDK compliant port/parameter inter will need to implement the body of the peripheral.	face. You
Import to XPS		
	Options Determine the setting of the set of t	
	Brows	e
More Info	< Back Next >	Cancel

Slika 5.1.3. Peripheral Flow strana

Na Repository or Project stranici, slika 5.1.4, treba odabrati gde će se sačuvati fajlovi periferije. Ukoliko se periferija koristi u više projekata treba odabrati opciju To an EDK user repository. Na ovaj način više različitih sistema može da importuje datu periferiju. Kako se u ovom primeru radi sa single embedded projektom treba odabrati opciju To an XPS project. Pošto smo CIP vizard pokrenuli iz XPS, lokacija direktorijuma je popunjena automatski. Kliknuti Next.

🔶 Create Peripheral	And the owner of the owner of the owner		? ×
Repository or Pro Indicate where	oject you want to store the new peripheral.		
A new peripheral ca XPS projects.	n be stored in an EDK repository, or in an XPS project. When stored in an EDK repository, the peripheral can	be acce	ssed by multiple
🔘 To an EDK us	er repository (Any directory outside of your EDK installation path)		
Repository:		-	Browse
To an XPS pro	bject		
Project:	C: \Users \Srdjan \Desktop \Master \provera \system	-	Browse
De de la coloridad			
C:\Users\Srdjan\	Jaced Under : Desktop  Master \provera \system \pcores		
More Info	< Back N	ext >	Cancel

Slika 5.1.4. Repository or Project strana

• Na *Name and Version* strani, slika 5.1.5 treba upisati ime i verziju periferije. U ovom primeru koristiće se ime *diode*. Podešavanja za verziju su ostavljena sa podrazumevanim vrednostima. Može se dodati i opis periferije u *Description* polju.

Name and Versio Indicate the na	<b>n</b> ame and version of ye	bur peripheral.	Ś
Enter the name of t	the peripheral (upper	case characters are not allowed). This name will be used as the t	op HDL design entity.
Name: diode			
Version: 1.00.a			
Major revision:	Minor revision:	Hardware/Software compatibility revision:	
1	00 🚔	a	
Logical library nar All HDL files (eith name above. An repositories indic	ne: diode_v1_00_a er created by you or y other referred logic ated in the XPS proje	generated by this tool) that are used to implement this periphera al libraries in your HDL are assumed to be available in the XPS pro ct settings.	l must be compiled into the logical library ect where this peripheral is used, or in EDK

Slika 5.1.5. Name and Version strana

- Na *Bus Interface* strani, slika 5.1.6 bira se procesorska magistrala na koju se povezuje periferija koju razvijamo. Na raspolaganju su:
  - Processor Local Bus (PLBv46) omogućava high-speed interfejs između procesora i periferije. Koristi se i kod MicroBlaze i kod PowerPC procesora.
  - Fast Simplex Link (FSL) je point-to-point FIFO interfejs. Može se koristiti sa MicroBlaze procesorom, dok se najčešće ne koristi sa PowerPC procesorima.

U ovom primeru se koristi PLB v46. Kliknuti Next.

Bus Interface Indicate the bus interface supported by your peripheral.	3
To which bus will this peripheral be attached?	
Processor Local Bus (PLB v4.6)	
Fast Simplex Link (FSL)	
ATTENTION Refer to the following documents to get a better understanding of how user p	peripherals connect to the CoreConnect(TM) bus PLB v4.6
ATTENTION Refer to the following documents to get a better understanding of how user p interconnect and the FSL interface.	peripherals connect to the CoreConnect(TM) bus PLB v4.6
ATTENTION Refer to the following documents to get a better understanding of how user p interconnect and the FSL interface. NOTE - Select the bus interface above and the corresponding link(s) will appea <u>CoreConnect Specification</u>	peripherals connect to the CoreConnect(TM) bus PLB v4.6 ar below for that interface.
ATTENTION Refer to the following documents to get a better understanding of how user p interconnect and the FSL interface. NOTE - Select the bus interface above and the corresponding link(s) will appea <u>CoreConnect Specification</u> PLB (v4.6) Slave IPIF Specification for single data beat transfer	peripherals connect to the CoreConnect(TM) bus PLB v4.6 ar below for that interface.
ATTENTION Refer to the following documents to get a better understanding of how user p interconnect and the FSL interface. NOTE - Select the bus interface above and the corresponding link(s) will appea <u>CoreConnect Specification</u> PLB (v4.6) Slave IPIF Specification for single data beat transfer PLB (v4.6) Slave IPIF Specification for burst data transfer	peripherals connect to the CoreConnect(TM) bus PLB v4.6 ar below for that interface.
ATTENTION Refer to the following documents to get a better understanding of how user p interconnect and the FSL interface. NOTE - Select the bus interface above and the corresponding link(s) will appea <u>CoreConnect Specification</u> PLB (v4.6) Slave IPIF Specification for single data beat transfer PLB (v4.6) Slave IPIF Specification for burst data transfer PLB (v4.6) Master IPIF Specification for single data beat transfer	peripherals connect to the CoreConnect(TM) bus PLB v4.6 ar below for that interface.
ATTENTION Refer to the following documents to get a better understanding of how user p interconnect and the FSL interface. NOTE - Select the bus interface above and the corresponding link(s) will appea <u>CoreConnect Specification</u> PLB (v4.6) Slave IPIF Specification for single data beat transfer PLB (v4.6) Master IPIF Specification for burst data transfer PLB (v4.6) Master IPIF Specification for burst data transfer PLB (v4.6) Master IPIF Specification for burst data transfer	peripherals connect to the CoreConnect(TM) bus PLB v4.6 ar below for that interface.

Slika 5.1.6. Bus Interface strana

Na IPIF strani, slika 5.1.7 su opcije koje su na raspolaganju korisniku, zavisno da li je nova periferija definisana kao master ili slave. Dodatno objašnjenje za svaku od ovih opcija može se dobiti klikom na More Info. U ovom primeru će se koristiti User logic software register i Include data phase timer. User logic software register su registri koji su dostupni koristeći memory map adrese u softverskom kodu. Registri su nam neophodni za smeštanje zaglavlja, ali i za prosleđivanje informacija koje diode treba upaliti. Include data phase timer je time-out funkcionalnost koju koriste sve periferije u dizajnu. Software reset blok daje mogućnost resetovanja kompletne periferije softverski. Read/write FIFO daje mogućnost implementacije Read/Write FIFO funkcionalnosti ka i od magistrale. Interrupt control opcija omogućava periferiji da generiše interrupt signale koji bi bili obrađeni u procesoru. U ovom primeru nam neće biti neophodne ove opcije.

🗇 Create Peripheral - IPIF (IP Interface) Services	
IPIF (IP Interface) Services Indicate the IPIF services required by your peripheral.	
Your peripheral will be connected to the PLB (v4.6) interconnect through correspondent the vitar look offers other commonly used services and configurations to sime vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other commonly used services and configurations to sime the vitar look offers other look other look offers other look offers othe	onding PLB IP Interface (IPIF) modules, which provide you with a quick way to s the standard functions like address decoding provided by the slave IPIF module, mplify the implementation of the design. Slave service and configuration Typically required by most peripherals for operations like logic control, status report, data buffering, multiple memory/address space access, and etc. (PLB slave interface will always be included). Software reset Read/Write FIFO Interrupt control Master service and configuration Typically required by complex peripherals like Ethernet and PCI for commanding data transfers between regions (PLB master interface will be included if master service selected). User logic master
More Info	< Back Next > Cancel

Slika 5.1.7. IPIF Services strana

• Na strani *User S/W registers*, slika 5.1.8 treba odabrati koliko registara se koristi. U ovom primeru se koristi 12 (maksimum je 4096).

Create Peripheral	? ×
User S/W Register Configure the software accessible registers in your peripheral	
The user specific software accessible registers will be implemented software programs to control and to monitor the status of your us or quad word boundaries depending on your design. An example k wizard tool for your reference.	in the user-logic module of your peripheral. Such registers are typically provided for er logic. These registers are addressable on the byte, half-word, word, double word ogic for register read/write will be included in the user-logic module generated by the
Bus2IP_RdReq Bus2IP_WReq Bus2IP_WrCE Bus2IP_Data IP2Bus_Data IP2Bus_RdAdk IP2Bus_Error User Logic	User logic software registers may take full advantage of the slave IPIF address- decoding service to generate CE decodes for all of the individual register of interest. The diagram on the left shows the simplest set of IPIC slave signals to read/write the registers. Number of software accessible registers: 10  10  10  10  10  10  10  10  10  1
More Info	< Back Next > Cancel

Slika 5.1.8. User S/W Register strana

• Na *IP Interconnect* strani, slika 5.1.9 je spisak IPIC signala koje CIP vizard nudi. Objašnjenja za ove signale se mogu pogledati ako se neki od njih odabere. Ostaviti podrazumevane vrednosti i kliknuti *Next*.

Create Peripheral      IP Interconnect (IPIC)     Select the interface between the logic to be imp      Your peripheral will be connected to the PLB (v4.6) in     module interfaces to the IPIF module(s) and other s	Nemented in your peripheral and the IPIF. Interconnect through suitable IPIF master/slave ub-blocks through a set of signals called the IP	module(s). Your custom logic from the user-logic nterconnect (IPIC) interface. Some of the ports are not then extinged parts the degrad
based on your needs.	Note: all IPIC ports are active high.          Image: State st	Port description
More Info	Restore Defaults	< Back Next > Cancel

Slika 5.1.9. IP Interconnect strana

- Na strani *Peripheral Simulation Support*, slika 5.1.10 može se odabrati opcija da CIP generiše BFM simulacionu platformu. Prethodno je potrebno instalirati:
  - BFM simulacioni paket za EDK
  - ModelSim-SE ili Model Sim-PE

U ovom primeru se neće koristiti ova opcija, tako da samo treba kliknuti Next.

CIP vizard kreira dva HDL fajla koji implementiraju radni okvir za pcore.

diode.vhd fajl, koji sadrži PLBv46 *bus* interfejs logiku. Kako će u ovom projektu portovi da se konektuju na "spoljašnji svet", potrebno je modifikovati imena portova i dati im odgovarajuća imena (primer dat u poglavlju 5.2 jasno ilustruje navedeno). U fajlu su precizna uputstva gde se dodaju imena portova.

 user\_logic.vhd fajl, koji je templejt fajl gde se dodaje logika koju periferija obavlja. Iako je uvek moguće kreirati dodatne *source* fajlove, u ovom primeru je dovoljan samo user\_logic.vhd fajl.

Create Peripheral (OPTIONAL) Peripheral Simulation Support Generate optional files for simulation using Bus Functional Models (BFM).	ि <u>×</u>
The EDK provides a BFM simulation platform to help you simulate your peripher Functional Language (BFL) stimulus file for the target bus.	ral. Indicate if you want this tool to generate the appropriate HDL and Bus  Generate BFM simulation platform for ModelSim-SE or ModelSim-PE This feature requires that you have accepted the associated IBM license agreement and installed the BFM package. The link below shows how: BFM Package Installation Instructions
More Info	< Back Next > Cancel

Slika 5.1.10. Peripheral Simulation Support strana

- *Peripheral Implementation Support* strana, slika 5.1.11 nudi tri opcije za kreiranje opcionih fajlova za hardversku i softversku implementaciju.
  - Prva opcija je da se user\_logic kreira u Verilogu umesto u VHDL.
  - Ukoliko je potrebno da se *pcore* dizajn implementira u potpunosti (za vremenske analize i simulacije), kliknuti na opciju *Generate ISE and XST files to help you implement the peripheral using XST flow.* CIP vizard kreira potrebne ISE *project* fajlove. Ukoliko je periferija *low-speed*, ovo nije neophodno.

- Ukoliko periferija zahteva kompleksnije softverske drajvere, klinkuti na *Generate template driver files to help you impement software interface.* 

U ovom primeru nećemo odabrati nijednu od ovih opcija. Kliknuti Next.

🗢 Create Peripheral		? X
(OPTIONAL) Peripheral Implementation Sup Generate optional files for hardware/software	<b>port</b> implementation	
Upon completion, this tool will create synthesizable for the synthesizable templates, so that you can be Peripheral (VHDL) IPIF (VHDL) User Logic (VHDL)	HDL files that implement the IPIF services you requested. A stub 'user_logic' module will be cristed using standard HDL design flows. The tool will also generate EDK interface files (mpd/ ook up the generated peripheral to a processor system. Note Should the peripheral interface (ports/parameters) or file list change, you will need to regenerate the EDK interface files using the import functionality of this tool. Generate stub 'user_logic' template in Verilog instead of VHDL Generate ISE and XST project files to help you implement the peripheral using XST flo Generate template driver files to help you implement software interface	N
More Info	< Back Next > C	ancel

Slika 5.1.11. Peripheral Implementation Support strana

Kad završimo sa podešavanjima, dobićemo *Summary report* stranu, slika 5.1.12. Kliknuti *Finish* da bi se završio proces generisanja periferije.

🔶 Create Peripheral	2 ×
	Congratulations!
	When you click Finish, HDL files representing your peripheral will be generated. You will have to implement the functionality of your peripheral in the stub 'user_logic' template file.
	IMPORTANT: If you make any interface changes to the generated peripheral (including peripheral name, version, ports and parameters), or any file changes (add or remove files), you will need to regenerate the EDK interface files by using this tool in the Import mode.
	Thank you for using Create and Import Peripheral Wizard! Please find your peripheral hardware templates under C: \Users\Srdjan\Desktop\Master\provera\system/pcores/diode_v1_00_a.
	Peripheral Summary:
Λ΄	top name : diode version : 1.00.a type : PLB (v4.6) slave
	features : slave attachment user s/w registers
1	Address Block Summary:
	user logic slv : C_BASEADDR + 0x00000000 : C_BASEADDR + 0x000000FF
	File Summary
	NOTE: A *.cip settings file will be created under your peripheral's "devl" folder. It can be loaded in a future wizard session to regenerate your peripheral.
	Click Finish to generate your peripheral.
More Info	< Back Finish Cancel

Slika 5.1.12. Summary report strana

## 5.2. Modifikacija CIP templejt fajla

Sledeći korak jeste modifikacija diode.vhd i user\_logic.vhd fajlova, kako bi se implementirala odgovarajuća funkcionalnost.

- U XPS alatu, kliknuti *File > Open*
- Naći *pcores/diode\_v1\_00\_a/hdl/vhdl* direktorijum i otvoriti diode.vhd fajl. Ovde treba dodati ulazne i izlazne portove na dva mesta:
  - Deklaracija portova top level entiteta, slika 5.2.1
  - Mapa portova za instancu user\_logic, slika 5.2.2

Skrolovati do linije 165 otprilike. U delu predviđenom za to treba ubaciti uokvireni kod sa slike.

![](_page_35_Figure_7.jpeg)

Slika 5.2.1. Dodavanje portova u diode.vhd fajlu

Takođe, skrolovati kursor do linije 390 otprilike. U odgovarajućem delu, ubaciti uokvireni deo koda.

390		port map	
391	¢	(	
392		MAP USER PORTS BELOW THIS LINE	
393		LEDs(7 downto 0)	=> LEDs(7 downto 0),
394		DIP_switch(7 downto 0)	=> DIP_switch(7 downto 0),
395	¢	USER ports mapped here	
396	-	MAP USER PORTS ABOVE THIS LINE	

Slika 5.2.2. Mapiranje portova u diode.vhd fajlu

Sačuvati promene u fajlu. Dodavanje korisničkih portova i njihovo mapiranje su najčešće izmene koje se vrše u *<ip core name>.vhd* fajlu.

Na isti način otvoriti i user\_logic.vhd fajl. Skrolovati do linije 99 otprilike, i ubaciti uokvireni kod sa slike 5.2.3 (ubacivanje deklaracije portova u deo gde se deklarišu portovi entiteta).

97		port	
98	¢	(	
99		ADD USER PORTS BELOW THIS LINE	
100		DIP_switch	: in std_logic_vector(7 downto 0);
101		LEDs	: out std_logic_vector(7 downto 0);
102	þ	USER ports added here	
103	-	ADD USER PORTS ABOVE THIS LINE	

![](_page_35_Figure_15.jpeg)

Dalje, u delu za deklaraciju signala, oko linije 132, ubaciti sledeće signale:

signal hdr1: std\_logic\_vector(159 downto 01011"; signal hdr2 : std\_logic\_vector(159 downto 01011"; signal hdr3 : std logic vector (159 downto 00111"; signal hdr4 : std logic vector (159 downto 00111"; signal hdr5 :

signal led\_out : std\_logic\_vector(7 downto 0);

signal hdr : std\_logic\_vector(159 downto 0);

Signal *led\_out* predstavlja vrednost na osnovu koje se pale odgovarajuće diode. Vrednost signala *header* zavisi od kombinacije na DIP sviču, i omogućava procesoru iščitavanje željene vrednosti. Ostali signali (*hdr1* do *hdr5*) su vrednosti zaglavlja koje ćemo koristiti u testiranju (u finalnoj implementaciji bi sadržaj zaglavlja bio upisan u registar zaglavlja iz ravni podataka). Na osnovu vrednosti prekidača na DIP sviču, u registar zaglavlja se upisuje sadržaj jednog od ovih pet signala. U ovom delu, korisnik može ubaciti i druga zaglavlja koja želi da testira. U okviru user\_logic.vhd fajla, definišu se i dva procesa. Prvi proces služi za iščitavanje podataka iz registara u zavisnosti od kombinacije na DIP sviču. Kako bi se kod uprostio, vrednost zaglavlja se dodeljuje signalu *hdr*. Na ovaj način će kod u procesu koji reguliše iščitavanje zaglavlja biti znatno jednostavniji nego da se u njemu vrši provera stanja na DIP sviču.

```
decode1: process(BUS2IP_Clk, Bus2IP_Reset, DIP_switch)
    begin
    if (BUS2IP_Clk 'event and BUS2IP_Clk = '1' and Bus2IP_Reset = '0') then
        if(DIP_switch="00000001")then
            hdr(31 downto 0)<=hdr2(31 downto 0);
            hdr(63 downto 32)<=hdr2(63 downto 32);
            hdr(95 downto 64)<=hdr2(95 downto 64);
            hdr(127 downto 96)<=hdr2(127 downto 96);
            hdr(159 downto 128)<=hdr2(159 downto 128);
        elsif (DIP_switch="00000010")then
            hdr(31 downto 0)<=hdr1(31 downto 0);</pre>
```

```
hdr(63 downto 32) <= hdr1(63 downto 32);
            hdr(95 downto 64) <= hdr1(95 downto 64);
            hdr(127 downto 96) <= hdr1(127 downto 96);
            hdr (159 downto 128) <= hdr1 (159 downto 128);
      elsif (DIP_switch="00000100")then
            hdr(31 downto 0) <= hdr3(31 downto 0);
            hdr(63 downto 32) <= hdr3(63 downto 32);
            hdr(95 downto 64) <= hdr3(95 downto 64);
            hdr (127 downto 96) <= hdr3 (127 downto 96);
            hdr (159 downto 128) <= hdr3 (159 downto 128);
      elsif (DIP_switch="00001000")then
            hdr(31 downto 0) <= hdr4(31 downto 0);
            hdr(63 downto 32) <= hdr4(63 downto 32);
            hdr (95 downto 64) <= hdr4 (95 downto 64);
            hdr(127 downto 96) <= hdr4(127 downto 96);
            hdr (159 downto 128) <= hdr4 (159 downto 128);
      elsif (DIP switch="00010000")then
            hdr(31 downto 0) <= hdr5(31 downto 0);
            hdr(63 downto 32) <= hdr5(63 downto 32);
            hdr (95 downto 64) <= hdr5 (95 downto 64);
            hdr(127 downto 96) <= hdr5(127 downto 96);
            hdr(159 downto 128) <= hdr5(159 downto 128);
      else
            hdr(159 downto 0) <= (others=>'1');
      end if;
end if;
end process decode1;
```

Drugi proces reguliše paljenje dioda. Četiri diode služe za prikaz broja validnih zaglavlja, a druge četiri za prikaz broja pogrešnih zaglavlja. Signal *leds\_out* upravlja diodama, pa njegova četiri bita se iščitavaju iz *slv\_reg0* (u kome su smešteni podaci o broju validnih zaglavlja), a druga četiri iz *slv\_reg1* (u kome su smešteni podaci o broju pogrešnih zaglavlja). Prethodno procesor upisuje odgovarajuće vrednosti u ove registre. Vrednost signala *leds\_out* se zatim dodeljuje izlaznom portu *LEDs* koji je vezan na pinove LED dioda (ovo će biti objašnjeno u odeljku 5.3.1).

```
decode2: process(BUS2IP_Clk, Bus2IP_Reset)
begin
    if (BUS2IP_Clk 'event and BUS2IP_Clk = '1' and Bus2IP_Reset = '0') then
        led_out(3 downto 0) <= slv_reg0(28 to 31);
        led_out(7 downto 4) <= slv_reg1(28 to 31);
        LEDs <=led_out;
    end if;
end process decode2;</pre>
```

Promene treba napraviti i u delu kojim se reguliše isčitavanje registara. Prikazani kod na slici 5.2.4 prikazuje registre vidljive procesoru. Promene postoje samo u delu vezanom za registre *slv\_reg2* do *slv\_reg6* jer procesor iz njih iščitavaju zaglavlja. U procesu *decode1* je definisano da se na osnovu stanja na DIP sviču podaci upisuju u signal *hdr*. Ovde se ti podaci koje sadrži signal *hdr* prosleđuju procesoru na obradu. Pronaći *SLAVE\_REG\_READ\_PROC*, i napraviti izmene prikazane na slici 5.2.4.

338 🛱	<pre>case slv_reg_read_sel is</pre>
339	<pre>when "10000000000" =&gt; slv_ip2bus_data &lt;= slv_reg0;</pre>
340	<pre>when "01000000000" =&gt; slv_ip2bus_data &lt;= slv_reg1;</pre>
341	<pre>when "001000000000" =&gt; slv_ip2bus_data &lt;= hdr(31 downto 0);</pre>
342	<pre>when "000100000000" =&gt; slv_ip2bus_data &lt;= hdr(63 downto 32);</pre>
343	<pre>when "000010000000" =&gt; slv_ip2bus_data &lt;= hdr(95 downto 64);</pre>
344	<pre>when "000001000000" =&gt; slv_ip2bus_data &lt;= hdr(127 downto 96);</pre>
345	<pre>when "000000100000" =&gt; slv_ip2bus_data &lt;= hdr(159 downto 128);</pre>
346	<pre>when "000000010000" =&gt; slv_ip2bus_data &lt;= slv_reg7;</pre>
347	<pre>when "000000000000" =&gt; slv_ip2bus_data &lt;= slv_reg8;</pre>
348	<pre>when "000000000100" =&gt; slv_ip2bus_data &lt;= slv_reg9;</pre>
349	<pre>when "000000000010" =&gt; slv_ip2bus_data &lt;= slv_reg10;</pre>
350	<pre>when "00000000001" =&gt; slv_ip2bus_data &lt;= slv_reg11;</pre>
351	when others => slv_ip2bus_data <= (others => '0');
352 -	end case;
353	
354 -	end process SLAVE REG READ PROC;

Slika 5.2.4. Izmene u kodu u user\_logic.vhd fajlu

### 5.3. Dodavanje kreirane periferije u Processor System

Modifikovani su diode.vhd i user\_logic.vhd fajlovi dodavanjem novih portova. Svaki put kada se promene ovi fajlovi u smislu da se menjaju portovi ili parametri u MPD fajlu, CIP vizard mora da se reimportuje. Tako se regenerišu PSF, MPD i PAO fajlovi, koji su interfejs fajlovi ka EDK. Kada se importovanje završi, nova periferija može biti dodata u *embedded* dizajn. Pre nego što se izvrši reimportovanje periferije, mali podsetnik šta je do sada urađeno:

- Prvi put kad je pokrenut CIP vizard, kreirana je periferija *diode*, podešeni *bus* interfejsi i generisani zahtevani templejt fajlovi.
- Sada treba dodati *diode* u postojeći projekat, opet koristeći CIP vizard. Periferija *Diode* je importovana u XPS-odgovarajući direktorijum, a CIP vizard kreira MPD i PAO fajlove.

Postupak je sledeći:

- Otvoriti CIP vizard i čekirati opciju import an existing peripheral to an XPS project.
- Na *Repository or Project* strani, odabrati postojeći projekat.
- Na strani *Name and Version*, izabrati diode, sa *Name drop-down* liste. Verzija se ne zahteva, ali je u ovom primeru već odabrana.
- Ako CIP vizard pita da li da izmeni postojeću periferiju sa ovim imenom, kliknuti *Yes*.
- Odabrati HDL source fajl, slika 5.3.1. Kliknuti Next.

Source File Types Indicate the types of files that make up your peripheral.	×
Indicate the types of files that make up your peripheral.	
✓ HDL source files (*.vhd, *.vhdl, *.v, *.vh)	
Netlist files (*.edn, *.edf, *.ngc, *.ngo)	
Documentation files (*.pdf, *.doc, *.txt)	
More Info	< Back Next > Cancel

Slika 5.3.1. Source File Types strana

- CIP vizard importuje *pcores* koji su bili kreirani na različite načine. Ukoliko je CIP vizard korišćen da se kreira periferija, najbolji način da se pravilno lociraju i identifikuju izvorišni fajlovi je da se koristi *Peripheral Analysis Order* (PAO) fajl. Odabrati opciju *Use existing Peripheral Analysis Order* fajl (\*.pao), slika 5.3.2.
- Pronaći odgovarajući PAO fajl. On je lociran u *diode/data* subdirektorijumu. Kada se koristi PAO fajl da se lociraju potrebni *source* fajlovi, nije neophodno dodavati nikakve dodatne fajlove ili biblioteke. Ukoliko se importuje kompleksnija periferija koja sadrži više fajlova, treba pogledati listing putanja biblioteka i HDL *source* fajlova, da bi se utvrdilo da li su svi potrebni fajlovi i biblioteke uključene.

Import Peripheral	? ×
HDL Source Files Indicate how this tool should locate the HDL files that make up your peripheral.	2
HDL language used to implement your peripheral: VHDL	
Use data (*.mpd) collected during a previous invocation of this tool	
	Browse
How to locate your HDL source files and dependent library files	
Use an XST project file (*.prj)	
This tool will input the HDL file-set and the logical libraries they are compiled into from the appropriate lines in the project file.	
	Browse
<ul> <li>Use existing Peripheral Analysis Order file (*.pao)</li> </ul>	
C:\Users\Srdjan\Desktop\Master\provera\system\pcores\diode_v1_00_a\data\diode_v2_1_0.pao	Browse
Browse to your HDL source and dependent library files (*.vhd, *.vhdl, *.v, *.vh) in next step	

Slika 5.3.2. HDL Source File strana

• Na *HDL Analysis Information* stranici, slika 5.3.3, treba proveriti da li su user\_logic.vhd i diode.vhd fajlovi na dnu liste. Ukoliko dva VHDL fajla koji su dodati nisu još uvek kompajlirani, kliknuti *Next* da se kompajliraju i otvoriti *Bus Interface* stranicu.

lse	the buttons on the	rig	nt to add and remov	/e files, indicate logica	libraries and set the HUL analyze order. New sub-HUL libraries will also be imported
	Language		Logical Library	DL Source File Pat	Add Files
1	vhdl		proc_common	C:\Xilinx\12.2\I	
2	vhdl		proc_common	C:\Xilinx\12.2\I	Add Library.
3	vhdl		proc_common	C:\Xilinx\12.2\I	
4	vhdl		proc_common	C:\Xilinx\12.2\I	
5	vhdl		proc_common	C:\Xilinx\12.2\I	
6	vhdl		proc_common	C:\Xilinx\12.2\I	
7	vhdl		proc_common	C:\Xilinx\12.2\I	
8	vhdl		plbv46_slave_si	C:\Xilinx\12.2\I	
9	vhdl		plbv46_slave_si	C:\Xilinx\12.2\I	
10	vhdl		plbv46_slave_si	C:\Xilinx\12.2\I	
11	vhdl	-	diode_v1_00_a 👻	C:\Users\Srdjan	Remove
12	vhdl	-	diode_v1_00_a 👻	C:\Users\Srdjan	Move Up
					Move Down

Slika 5.3.3. HDL Analysis Information strana

• Odabrati odgovarajući *bus* interfejs, slika 5.3.4. Periferija *diode* koristi *PLBv46 Slave* interfejs.

Bus Interfaces Identify the bus interfaces supported by your peripheral.	3
A bus interface is a group of related interface ports distinguish by your peripheral or indicate if there is no applicable bus inter	hed by a bus standard (i.e. PLBv46, DCR, or FSL). Select the bus interface(s) supported rface.
Select bus interface(s)	
Processor Local Bus (version 4.6) interface	Fast Simplex Link bus interface
PLBV46 Master (MPLB)	FSL Master (MFSL)
Generate burst	FSL Slave (SFSL)
V PLBV46 Slave (SPLB)	
Device Control Register bus interface	
DCR Slave (SDCR)	

Slika 5.3.4. Bus Interfaces strana

• Na *SPLB:Port* stranici, slika 5.3.5 je prikazan kompletan listing svih PLBv46 *bus* signala koji su korišćeni u dizajnu. Dat je spisak signala i pridruženih *bus* protokola koje je CIP vizard automatski konfigurisao. Ukoliko je periferija kreirana pomoću CIP vizarda, svi neophodni signali su uključeni. Ukoliko je periferija kreirana pomoću nekog drugog alata ili sadrži kompleksne *bus* interfejse, ova strana je korisna za analiziranje *bus* signala.

PL	<b>B : Port</b> Define the SPLB bus i	nterface port(s) for this peripheral.		3
he iuto Bus	SPLB bus interface is matically done the se Interface Port(s): SPL	defined by a predefined set of ports and par ections for you. Otherwise indicate the port B	ameters. If your peripheral follows the standard naming conv s that correspond to the bus connectors.	entions, this tool has
	PLB Bus Connecto	Your Port	ATTENTION	
1	SPLB_CIk	SPLB_CIk	The Wizard has successful interface ports for SPI B by	ly extracted bus
2	SPLB_Rst	SPLB_Rst	naming convention.	opprysing eigenes
3	PLB_abort	PLB_abort		
4	PLB_ABus	PLB_ABus		
5	PLB_UABus	PLB_UABus		
6	PLB_BE	PLB_BE	-	

Slika 5.3.4. SPLB : Port strana

• Na *SPLB:Parameter* stranici, slika 5.3.5 periferija *diode* je mapirana u jedan adresni opseg, koji XPS bira kad je *pcore* uključen u dizajn. Kompleksnije periferije mogu takođe sadržati memorijske blokove koji moraju biti dostupni. Kao *high adress* odabrati *C\_HIGHADRR*. Kliknuti *Next*.

Import Peripheral		_	State State		<u> ?</u>
SPLB : Parameter Define the SPLB bus interface p	parameter(s) for this periphera	al.			×.
The SPLB bus interface is defined b automatically done the selections fo	y a predefined set of ports an or you. Otherwise check off th	id parameter ne values.	s. If your peripheral follo	ows the standard naming c	onventions, this tool has
Register Space					
Parameter determine base addre	ss:	[	C_BASEADDR		•
Parameter determine high addres	ss:	[	C_HIGHADDR		•
Memory Space					
e Address Parame h Address	Paramı Cacheable				Add
					Remove
					_
More Info				< Back No	ext > Cancel

Slika 5.3.5. SPLB : Parameter strana

- Na sledećoj stranici ostaviti podrazumevana podešavanja i kliknuti Next.
- Periferija *diode* ne sadrži *interrupt* signale. Deselektovati opciju *Select and configure interrupts*, slika 5.3.6 i kliknuti *Next* da bi se otvorila *Parameter Attribute* stranica.

dentify Interrupt Signals Identify the interrupt signals on your peripheral.	
ndicate the attributes of the interrupt signals by checking the isses this information to automatically connect the interrupt por	interrupt port name on the left and then clicking on the radio buttons to the right. EDK ts of your peripheral.
DIP_switch	Interrupt sensitivity of port:
	Falling edge sensitive  Low level sensitive
	Rising edge sensitive High level sensitive

Slika 5.3.6. Identify Interrupt Signals strana

• Na stranici *Port Atributes* su dostupne detaljnije informacije o dodatim portovima. Klinuti *Next*.

arameter Attributes					
Identify the parameters that re	quire spe	cial handling.			÷.
	1.00 - 11				
he system it is instantiated in.	a fill in the	attribute values to the righ	t. These attribu	ites help the various tools ir	n EDK to integrate this peripheral into
- List User Parameters only -	•	Attributes:			
C_INCLUDE_DPHASE_TIMER			1		
		Parameter Name			
		Data Type			
		Default Value			
		Display advanced at	tributes		

Slika 5.3.6. Parameter Attributes strana

• Konačno, prikazuje se stranica sa detaljima projekta, slika 5.3.7. Klinuti *Finish* kako bi se proces importovanja kompletirao.

Timport Peripheral	
	Congratulations!
	Your peripheral will now be added to the current XPS project. You can now instantiate this peripheral in your system just as you instantiate other peripherals.
	Thank you for using Create and Import Peripheral Wizard! Please find your imported peripheral under C: \Users\Srdjan\Desktop\Master\provera\system\pcores\diode_v1_00_a.
	Summary:
	Logical library : diode_v1_00_a Version : 1.00.a Bus interface(s) : SPLB
	The following sub-directories will be created:
	<pre>- diode_v1_00_a\data - diode_v1_00_a\hdl - diode_v1_00_a\hdl\vhdl</pre>
1	The following HDL source files will be copied into the diode_v1_00_a\hdl\vhdl directory:
	- user_logic.vhd - diode.vhd
	The following files will be created under the diode_v1_00_a\data directory:
	- diode_v2_1_0.mpd - diode_v2_1_0.pao ~
	☑ Save previously generated files
More Info	< Back Finish Cancel

Slika 5.3.7. Summary strana

#### 5.3.1. Dodavanje periferije diode u projekat

U XPS alatu, u *Project Information* delu, u *IP Catalog* tabu može se videti periferija *diode* u delu *Project Local pcores/User*. Pre nego što se periferija *diode* doda u dizajn, moraju se napraviti promene u postojećem dizajnu. LED diode i DIP svičevi su trenutno nakačeni na GPIO izlaze. Pošto se njima sada upravlja pomoću nove periferije, *LEDs\_8Bit i DIP\_Switches pcore* moraju da budu uklonjeni iz dizajna.

• U System Assembly View desni klik na LEDs\_8Bit i DIP\_Switches, i odabrati opciju Delete Instance, slika 5.3.1.1.

DIP_Switche LEDs_8Bit	Configure IP
Push_Butto clock_gener	View MPD View IP Modifications (Change Log) View PDF Datasheet Browse HDL Sources
	Driver: gpio_v3_00_a
	Delete Instance
	Make This IP Local

Slika 5.3.1.1. Brisanje instance u XPS alatu

• Pojaviće se dijalog prozor. Odabrati opciju Delete instance and its ports (both internal and external), slika 5.3.1.2.

🗢 Delete IP Instance
Do you want to delete instance DIP_Switches_8Bit and its associated ports?
<ul> <li>Delete instance and its ports (both internal and external)</li> </ul>
O Delete instance and its external ports
O Delete instance and its internal ports
O Delete instance but keep its ports
OK Cancel

Slika 5.3.1.2. Brisanje instance i portova

• Locirati *diode* u *IP Catalog* tabu, desni klik, i odabrati opciju *Add IP*, slika 5.3.1.3. U dijalog prozoru kliknuti OK. XPS dodaje IP u *System Assembly View*, i može se videti u *Bus Interface* tabu.

![](_page_48_Figure_6.jpeg)

Slika 5.3.1.3. Dodavanje periferije u dizajn

• Kliknuti na + pored imena periferije, otvoriti *drop-down* meni i odabrati opciju *mb\_plb* kako bi se povezali na PLB *bus*, slika 5.3.1.4.

<u>ک</u>	t <u>+</u> mdm_U		Ĥ.	mdm	1.00.g
	🖨 diode_0		Ð	diode	1.00.a
	SPLB	No Connectior 🖵			
	proc_sys_re	No Connection	π.	proc_sys_re	2.00.a
<u>.</u>	Push_Butto	New Connection	*	xps_gpio	2.00.a
	clock_gener	mb_plb	$\mathbf{\dot{\pi}}$	clock_gene	4.00.a

Slika 5.3.1.4. Povezivanje periferije na PLB magistralu

• Otvoriti *Ports* tab, proširiti diode\_0, odabrati *Make External* iz *drop-down* menija u *Net* koloni, slika 5.3.1.5. Podrazumevana imena su *diode\_0\_LEDs\_pin* i *diode\_0\_DIP\_switch*. Ime se može promeniti klikom na opciju *Name* u *Net* koloni, respektivno.

![](_page_49_Picture_1.jpeg)

Slika 5.3.1.5. Povezivanje portova na izlaz

- Otvoriti *Adresses* tab. *Diode\_0* bi trebalo da se nalaze u delu *Unmapped Addresses*, slika 5.3.1.6. Ukoliko nije tu, kliknuti *Project* > *Rescan User Repositories*.
- Kliknuti na *Generate Addresses* kako bi se periferiji dodelio odgovarajući adresni opseg.

Bus Interfaces Ports Ad	dresses							Generate Address
Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock	Generate Addresses
ighthere microblaze_0's Address Map								Generate Addresses
dlmb_cntlr	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB	dlmb		
ilmb_cntlr	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB	ilmb		
Push_Buttons_5Bit	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb_plb		
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb		
Unmapped Addresses								
diode_0	C_BASEADDR			U	SPLB	mb_plb		

Slika 5.3.1.6. Generisanje adresnog prostora za periferiju

- Konačno treba napraviti izmene i u system.ucf fajlu. Otvoriti projekat u ISE alatu, kliknuti na opciju *File > Open*, i odabrati *provera/data/system.ucf* fajl. Pronaći *fpga\_0\_LEDs\_8Bit\_GPIO\_IO\_0* i *fpga\_0\_DIP\_Switches\_GPIO\_IO\_0*. Dodele pinova su ostale u ucf fajlu, iako su periferije prethodno izbrisane. Bitno je zapamtiti da uklanjanje periferije ne znači automatski i ažuriranje UCF fajla.
- Zameniti fpga\_0\_LEDs\_8Bit\_GPIO\_IO\_0\_pin sa diode\_0\_LEDs\_pin na svih osam lokacija. Takođe, fpga\_0\_DIP\_Switches\_GPIO\_IO\_0 zameniti sa diode\_0\_DIP\_Switch\_pin i sačuvati UCF fajl, slika 5.3.1.7.

1	# Virtex 5 ML507 Evaluation Platform
2	Net diode_0_LEDs_pin<0> LOC = AE24   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
в	Net diode_0_LEDs_pin<1> LOC = AD24   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
1	Net diode_0_LEDs_pin<2> LOC = AD25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
5	Net diode_0_LEDs_pin<3> LOC = G16   IOSTANDARD=LVCMOS25   PULLDOWN   SLEW=SLOW   DRIVE=2;
6	Net diode_0_LEDs_pin<4> LOC = AD26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
7	Net diode_0_LEDs_pin<5> LOC = G15   IOSTANDARD=LVCMOS25   PULLDOWN   SLEW=SLOW   DRIVE=2;
в	Net diode_0_LEDs_pin<6> LOC = L18   IOSTANDARD=LVCMOS25   PULLDOWN   SLEW=SLOW   DRIVE=2;
Э	Net diode 0 LEDs pin<7> LOC = H18   IOSTANDARD=LVCMOS25   PULLDOWN   SLEW=SLOW   DRIVE=2;
10	Net fpga_0_Push_Buttons_5Bit_GPIO_IO_pin<0> LOC = AJ6   IOSTANDARD=LVCMOS33   PULLDOWN   SLEW=SLOW   DRIVE=2;
11	Net fpga 0 Push Buttons SBit GPIO IO pin<1> LOC = AJ7   IOSTANDARD=LVCMOS33   PULLDOWN   SLEW=SLOW   DRIVE=2;
12	Net fpga 0 Push Buttons Shit GPIO 10 pin<2> LOC = V8   105TANDARD=LVCROS33   PULLDOWN   SLEW=SLOW   DRIVE=2;
13	Net rpga 0 Push Buttons Shit GP10 10 pint3> LOC = AK/   IOSTANDARD=LVCMOS33   PULLDOWN   SLEW=SLOW   DRIVE=2;
14	NET FDGA U PUSH BUTTONS SEIT GPIO IO BINCAS LOC = US   IOSIANDARDELVCMOS33   PULLDOWN   SLEWESLOW   DRIVE=2;
15	Net didde - DIF Switch Dirk(V) LOC-023   IOSIANDARD-EVENUSIS   PULLDOWN   SLEW-SLOW   DRIVE-2;
17	Net didde_o_bir_Switch_pinkiy_boc-Ad2/   TOSTANDARD-Evenosis   Forbbown   Shew-Show   DRIVE-2,
L /	Net diode 0 DIE Switch pipe2> LOC=AF25   TOSTANDAPD=LVCMOS18   PULLDOWN   SIEW=SLOW   DETVE=2.
	Net diode 0 DIP_Switch_pin<2> LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
18	Net diode 0 DIP_Switch_pin<2> LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode 0 DIP_Switch_pin<3> LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode 0 DIP_Switch_pin<4> LOC=AF27   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
18 19 20	Net diode_0_DIP_Switch_pin<2> LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin<3> LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin<4> LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin<5> LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;
18 19 20 21	<pre>Net diode_0_DIP_Switch_pin&lt;2&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;3&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;4&gt; LOC=AF27   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2;</pre>
18 19 20 21 22	Net diode_0_DIP_Switch_pin<2> LOC=AF25       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<3> LOC=AF26       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<4> LOC=AF27       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<5> LOC=AF26       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<5> LOC=AF26       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<6> LOC=AF26       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<7> LOC=AF24       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;         Net diode_0_DIP_Switch_pin<7> LOC=AF24       IOSTANDARD=LVCMOS18       PULLDOWN       SLEW=SLOW       DRIVE=2;
18 19 20 21 22 23	<pre>Net diode_0_DIP_Switch_pin&lt;2&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;3&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;4&gt; LOC=AF27   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;7&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;7&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;7&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net fipqa 0_Clk 1 sys Clk pin TNM NET = sys Clk pin;</pre>
18 19 20 21 22 23 24	<pre>Net diode_0_DIP_Switch_pin&lt;2&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;3&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;4&gt; LOC=AF27   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;7&gt; LOC=AF24   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net fpqa_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin; TIMESPEC TS sys_clk_pin = PERIOD sys_clk pin 100000 kHz;</pre>
18 19 20 21 22 23 24 25	<pre>Net diode_0_DIP_Switch_pin&lt;2&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;3&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;4&gt; LOC=AF27   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;7&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net fpga_0_clk_1_sys_clk_pin TMM_NET = sys_clk_pin; TIMESPEC TS_sys_clk_pin = PERIOD_sys_clk_pin 100000 kHz; Net fpga_0_clk_1_sys_clk_pin LOC = AH15   IOSTANDARD=LVCMOS33;</pre>
18 19 20 21 22 23 24 25 26	<pre>Net diode_0_DIP_Switch_pin&lt;2&gt; LOC=AF25   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;3&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;4&gt; LOC=AF27   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;5&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;6&gt; LOC=AF26   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net diode_0_DIP_Switch_pin&lt;7&gt; LOC=AF24   IOSTANDARD=LVCMOS18   PULLDOWN   SLEW=SLOW   DRIVE=2; Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin; TIMESPEC TS_sys_clk_pin = PERIOD_sys_clk_pin 100000 kHz; Net fpga_0_clk_1_sys_clk_pin LOC = AH15   IOSTANDARD=LVCMOS33; Net fpga_0_rst_1_sys_rst_pin TIG;</pre>

Slika 5.3.1.7. Izmene koje je potrebno napraviti u system.ucf fajlu

Ukoliko se u diode.vhd ili user\_logic.vhd fajlovima vrše neke izmene u kodu (kao npr. izmena funkcionalnosti i sl.) nakon importovanja periferije, potrebno je ponovo generisati *bitstream* fajl. To se radi na sledeći način.

- U XPS alatu se naprave izmene u kodu i klikne na *Save*. Nakon toga se klikne na *Project > Clean all generated files*
- Kad se taj proces završi klikne se na Hardware > Generate Netlist
- Nakon toga se u ISE alatu klikne na Generate Programming File

#### 5.3.2. Eksportovanje dizajna i generisanje novog bitstream fajla

Sledeći korak je da se hardverski dizajn eksportuje i da se generiše novi *bitstream* fajl. Nakon toga, treba napisati aplikaciju u C, i dizajn je spreman za testiranje.

- U XPS alatu, odabrati *Project > Export Hardware Design to SDK*. Prihvatiti podrazumevani direktorijum i kliknuti *Export Only*.
- Kad se proces eksportovanja završi, zatvoriti XPS, vratiti se u ISE, i dvostruki klik na *Generate Programming File*.
- Kad se završi ovaj proces (obično traje par minuta), otvoriti SDK i kreirati novi radni prostor (*workspace*), slika 5.3.2.1. Kad se SDK otvori, potrebno je označiti eksportovani system.xml fajl.
- Odabrati *File > New > Xilinx Hardware Platform Specification* za kreiranje novog hardverskog projekta.
- Dati proizvoljno ime projektu (ovde je korišćeno *led\_diode*) i označiti system.xml fajl, slika 5.3.2.2. Dotični xml fajl bi trebalo da je smešten u *system/SDK/SDK\_Export/HW* folderu projekta.
- Kliknuti *Finish*.

Workspace	Launcher	×					
Select a workspace							
Xilinx SDK sto Choose a wo	ores your projects in a folder called a workspace. orkspace folder to use for this session.						
Workspace:	C:\Users\Srdjan\Desktop\Master\workspace	- Browse					
🔲 Use this as	; the default and do not ask again	OK Cancel					

Slika 5.3.2.1. Definisanje workspace foldera u SDK alatu

New Hardware Project	
New Hardware Project	:==
Create a new Hardware Project.	E
Project name: led_diode	
✓ Use default location	
Location: C:\Users\Srdjan\Desktop\Master\workspace\led_diode	Browse
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS projec The specification file and associated bitstream content will be copied into the	ct location. : workspace.
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS project The specification file and associated bitstream content will be copied into the C:\Users\Srdjan\Desktop\Master\provera\system\SDK\SDK_Export\hw\system	ct location. workspace. m.xml Browse
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS project The specification file and associated bitstream content will be copied into the C:\Users\Srdjan\Desktop\Master\provera\system\SDK\SDK_Export\hw\system Bitstream and BMM Files	ct location. e workspace. m.xml Browse
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS project The specification file and associated bitstream content will be copied into the C:\Users\Srdjan\Desktop\Master\provera\system\SDK\SDK_Export\hw\system Bitstream and BMM Files	ct location. e workspace. m.xml Browse
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS project The specification file and associated bitstream content will be copied into the C:\Users\Srdjan\Desktop\Master\provera\system\SDK\SDK_Export\hw\syste Bitstream and BMM Files	ct location. workspace. m.xml Browse
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS project The specification file and associated bitstream content will be copied into the C:\Users\Srdjan\Desktop\Master\provera\system\SDK\SDK_Export\hw\syste > Bitstream and BMM Files	ct location. workspace. m.xml Browse
Target Hardware Specification Provide the path to the hardware specification file exported from XPS. This file usually resides in SDK/SDK_Export/hw folder relative to the XPS project The specification file and associated bitstream content will be copied into the C:\Users\Srdjan\Desktop\Master\provera\system\SDK\SDK_Export\hw\syste	ct location. e workspace. m.xml Browse Cancel

Slika 5.3.2.2. Kreiranje novog projekta

#### i) Razvoj softvera za kreiranu periferiju

SDK otvara C/C++ *Perspective* sa tabelom u kojoj su prikazane sve periferije u dizajnu. Proveriti da li je *diode\_0* na listi. Postupak razvoja aplikacije je sledeći:

- Kreirati novi Xilinx C projekat, klikom na File > New > Xilinx C Project. Koristiti Hello World templejt, i ostaviti podrazumevano ime koje je predložio SDK. Korišćenje templejta je korisno, naročito za početnike, jer je na taj način već urađena većina podešavanja drajvera i slično. Napredniji korisnici, mogu kasnije ova podešavanja vršiti i ručno. Kliknuti Next.
- Podrazumevano podešavanje na ovaj strani je da SDK kreira novi BSP paket. Ostaviti tako. Kliknuti *Finish* da bi se generisao projekat.
- U *Project Explorer* kliknuti desni klik na novi projekat i odabrati *New* > *Source File*, slika 5.3.2.3.

<ul> <li>         Image: Bello_world_0</li></ul>	Modify	this BSP's Sett	ings	5
⊿ ≥ src ▶ C helloworld.	New	formation +	1	Project
<ul> <li>in platform_cc</li> <li>in platform.c</li> <li>in platform.h</li> <li>in script.ld</li> <li>in BSP Documenta</li> <li>in BSP Documenta</li> <li>in incroblaze_0</li> <li>inibgen.log</li> <li>inibgen.options</li> <li>Makefile</li> </ul>	Open Open With Copy Paste Delete Move Rename	► Ctrl+C Ctrl+V Delete F2		File File from Template Folder Class Header File Source File Source Folder
in system.mss ▷ I led diode	Import Export			C Project C++ Project
	\$ Refresh	F5		Other Ctrl+N
	Exclude from build Index Build Configurations Make Targets Clean Selected File(s) Build Selected File(s) Team Compare With	> > >	ard	Support Console Lonsole Lonsole Properties T Properties T Untime libraries by runr S\common\bin\nt64\vcredi e adminstrative privileg ello_world_0.elf.elfchec
□ <sup>◆</sup> 💽 /hello_wo	Replace With Properties	Alt+Enter		

Slika 5.3.2.3. Dodavanje novog source fajla u projekat

- Staviti sledeća podešavanja:
  - Source Folder: hello\_world\_0/src
  - Source File: leds.c
  - *Template: Default C source template*

#### Kliknuti Finish.

*leds.c* fajl se otvara u SDK prozoru, i tu treba da se unese C kod, koji procesor izvršava. U prvom delu koda je *while/do* petlja koja omogućava izvršavanje koda kad se pritisne neko *push* dugme. Na DIP sviču je prethodno izabrana kombinacija na osnovu koje se učitava odgovarajuće zaglavlje (ovaj deo je definisan u user\_logic.vhd fajlu). Dalje se vrši provera zaglavlja. Zaglavlje se deli na blokove od po 16 bita. Zatim se ti blokovi sabiraju i ukoliko je zbir jednak 0xFFFF čeksuma je u redu. Osim toga treba utvrditi ispravnost polja verzije i ttl. Ukoliko je zaglavlje validno, ttl polje se umanjuje za 1 i vrši se ažuriranje zaglavlja koje se potom upisuje u odgovarajuće registre u periferiji. Takođe, i brojač corr (broj validnih zaglavlja) kojim se reguliše paljenje dioda se uvećava za jedan. Ukoliko je zaglavlje pogrešno uvećava se brojač uncorr koji takođe reguliše paljenje dioda. Četiri diode na ploči služe za brojanje validnih zaglavlja, a četiri za brojanje pogrešnih. Ove informacije (o broju tačnih i pogrešnih zaglavlja) se zatim smeštaju u *slv\_reg0* i *slv\_reg1*. Adrese ovih registara su XPAR\_DIODE\_0\_BASEADDR i XPAR\_DIODE\_0\_BASEADDR+1. Na kraju koda je još jedna while petlja. Ona se izvršava dok se push dugme ne vrati u nepritisnuto stanje kako bi se izbegla višestruka provera i ažuriranje istog zaglavlja. Kao što se vidi kod je veoma jednostavan tako da je znatno lakše napisati softverski kod za proveru i ažuriranje zaglavlja nego napisati kod koji opisuje hardver koji obavlja istu funkciju.

```
#include "xparameters.h"
#include "mb_interface.h"
#include "stdio.h"
#include "xgpio.h"
#include "xutil.h"
#include "stdlib.h"
int main() {
      int i, x1, x2, x[5], data, *addr_ptr, *addr_ptr1, y[10], Y1=0, Y2=0,
      SUMA=0, check=0, cor=0, uncor=0;
      do
      Ł
            do
            Ł
                  addr ptr = XPAR PUSH BUTTONS 5BIT BASEADDR;
                  data=*addr ptr;// ucitavanje podataka sa push button-a
            }while(data==0);// kod se izvrsava dalje kad se preko push button-a
unese bilo koja vrijednost razlicita od , tj kada se pritisne neki od push
button-a
            addr_ptr1=XPAR_DIODE_0_BASEADDR;//ucitavanje podataka iz periferije
            SUMA=0x0000000;//postavljanje sume na 0
            check=0;
            for(i=0; i<5; i++)</pre>
            ł
                  y[2*i]=((*(addr_ptr1+4-i+2))>>16)&0x0000FFFF;//ucitavanje
```

```
y[2*i+1]=(*(addr_ptr1+4-i+2))&0x0000FFFF;//blokova od po 16
bita
                   SUMA=SUMA+y[2*i]+y[2*i+1];// sabiranje dobijenih blokova
podataka
             }
            if((y[0]>>12)!=4 || y[4]>>8==0)// provera polja verzije i TTL
                   check=1;
            Y1=SUMA&0x0000FFFF;//racunanje check sume
            Y2=(SUMA>>16) &0x0000FFFF;
            SUMA=Y1+Y2;
            if(SUMA!=0x0000FFFF)// provera check sume
                   check=1;//greska
            if(check==0)//ukoliko je zaglavlje validno, uvecava se brojac,
zaglavlje se ažurira i upisuje u odgovarajuce registre
             Ł
                   cor++;
                   *(addr_ptr1) = cor;
                   x1=(y[4]>>8) &0x00000FF;
                   x1=x1-1;
                   x2=(y[4]) \& 0 x 0 0 0 0 0 0 FF;
                   y[4] = ((x1 << 8) | x2);
                   SUMA=0;
                   for(i=0; i<10; i++)</pre>
                         if(i!=5)
                                SUMA=SUMA+y[i];
                   Y1=SUMA&0x0000FFFF;
                   Y2=(SUMA>>16) & 0x0000FFFF;
                   SUMA=Y1+Y2;
                   y[5]=0xFFFFFFFF-SUMA;
                   for(i=0; i<5; i++)</pre>
                         *(addr_ptr1+7+i) = ((y[2*i]<<16)|(y[2*i+1]));</pre>
            }
            else//ako je zaglavlje pogrešno, uvecava se odgovarajuci brojac
                   uncor++;
                   *(addr_ptr1+1) = uncor;
             }
            do//petlja za kašnjenje
             Ł
                   addr ptr = XPAR PUSH BUTTONS 5BIT BASEADDR;
                   data=*addr_ptr;// ucitavanje podataka sa push button-a
             }while(data!=0);// kod se izvrsava dalje kad se preko push button-a
unese bilo koja vrijednost razlicita od 0
      }while(1);
}
```

• Kliknuti *Save*, i SDK će automatski kompajlirati projekat. SDK će prijaviti grešku zbog toga što u projektu postoji više *main()* funkcija. Treba izbrisati *hello\_world.c* i *platform.c*. Desni klik na ove fajlove, i izabrati *Delete*, slika 5.3.2.4. Sada bi projekat trebalo da se kompajlira uspešno.

陷 Project Explorer 🛛			system.xml	🙀 system
		□ 🔄 🏹		SUMA=0
<ul> <li>hello_world_0</li> <li>hello_udes</li> <li>helloworld_0</li> <li>helloworld.c</li> <li>helloworld.c</li> <li>helloworld.c</li> <li>helloworld.c</li> <li>helloworld.c</li> <li>helloworld.c</li> </ul>		New Open Open With		for (i= if Y1=SUM Y2= (SU Y 0: =
<ul> <li>h platform.h</li> <li>k platform.h</li> <li>k lscript.ld</li> <li>hello_world_bsp_0</li> <li>i BSP Documentation</li> <li>microblaze 0</li> </ul>		Copy Paste Delete		Ctrl+C Ctrl+V Delete ir
<ul> <li>Incrobate_0</li> <li>libgen.log</li> <li>libgen.options</li> <li>Makefile</li> <li>system.mss</li> <li>led_diode</li> </ul>	2 2	Move Rename Import Export		F2

Slika 5.3.2.4. Brisanje C fajla

# 6. VERIFIKACIJA DIZAJNA

Dakle, sada je završen razvoj hardverskog i softverskog dela projekta. U ISE alatu je prvo kreiran projekat. Dodali smo *source* fajl u vidu *embedded* procesora i u BSB konfigurisali hardverske parametre: tip procesora, veličinu RAM memorije, periferije itd. Kreirana je nova periferija koja vrši proveru i ažuriranje IPv4 zaglavlja, i nakon toga importovana u dizajn. Konačno, u SDK alatu je napisana odgovarajuća C aplikacija koju procesor izvršava i koja vrši zadatu funkciju. Na kraju preostaje verifikacija dizajna. U ovom primeru je korišćena *Virtex ML507* ploča. Na DIP sviču, korisnik bira odgovarajuću kombinaciju. U diode.vhd fajlu se dekodovanjem stanja DIP sviča bira odgovarajuće zaglavlje za ispitivanje. Pritiskom na *Push* dugme, pokreće se izvršavanje C koda. Procesor na osnovu kombinacije sa DIP sviča, učitava odgovarajuće zaglavlje iz registara u periferiji, i vrši proveru. Ukoliko je tačno, zaglavlje se ažurira i smešta u registre periferije koju smo importovali. Takođe, pale se odgovarajuće diode. Na osnovu upaljenih LED dioda može se prebrojati koliko je tačnih odnosno pogrešnih zaglavlja obrađeno. Spuštanje dizajna na ploču se vrši na sledeći način:

• U SDK otvoriti *Xilinx Tools > Program FPGA*, slika 6.1.

![](_page_56_Picture_3.jpeg)

Slika 6.1. Spuštanje dizajna na ploču

- Treba ubaciti putanju do *bitstream* (\*.bit) i *block memory map* (\*.bmm) fajlova koji se nalaze u *provera* folderu, slika 6.2.
- U polju "*ELF File to Initialize in Block RAM*" odabrati elf fajl, slika 6.2.
- Kliknuti na Program

Program FPGA						
Program FPGA						
specity the	Ditstrea	im and the ELF files that reside in BRAIN memory	40-0			
Hardware	onfigu					
Hardware Specification: C:\Users\Srdjan\Desktop\Master\workspace\led_diode\system.xml						
Bitstream:	C:\Users\Srdjan\Desktop\Master\provera\system.bit Browse.					
BMM File:	C:\Use	C:\Users\Srdjan\Desktop\Master\provera\edkBmmFile_bd.bmm				
Software Configuration						
Processor		ELF File to Initialize in Block RAM				
microblaze_0		C:\Users\Srdjan\Desktop\Master\workspace\hello_w				
?		Program	Cancel			

Slika 6.2. Podešavanje BIT, BMM i ELF fajlova

Postoji nekoliko zaglavlja, pri čemu je samo jedno zaglavlje korektno. Ono se iščitava ukoliko je na DIP sviču aktivna kombinacija 00000100. Druge kombinacije iščitavaju pogrešna zaglavlja, što bi trebalo da se vidi paljenjem odgovarajućih dioda. Testiranje dizajna na *ML507* razvojnoj ploči je potvrdilo ispravnost rada dizajna. U tabeli 6.3 je prikazana zauzetost resursa FPGA čipa. Iz tabele se vidi da se koristi mala količina resursa kao i da je zadovoljeno vremensko ograničenje za takt procesora navedeno u tabeli 4.2.6.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1875	44800	4%
Number of Slice LUTs	2019	44800	4%
NumberofBlockRAM/FIFO	16	148	10%
Total Memory used (KB)	576	5328	10%
Number of BUFG/BUFGCTRLs	2	32	6%
Number of BSCANs	1	4	25%
Number of DSP48Es	3	128	2%
NumberofPLL_ADVs	1	6	16%
Average Fanout of Non-Clock Nets	4.20		

Tabela 6.3 Zauzetost FPGA resursa

# 7. Zaključak

U ovom master radu je predstavljen proces dizajniranja embedded sistema koji vrši funkciju provere i ažuriranja IPv4 zaglavlja. Ovo je jedna od funkcija koja se izvršava u ravni podataka rutera. Ova implementacija se razlikuje po tome što nije čisto hardverska, već predstavlja kombinaciju hardverskog i softverskog rešenja. Generalno, razvoj ovakvih sistema je vrlo komplikovan jer zahteva dizajn i hardvera i softvera kao i njihovo međusobno integrisanje. Međutim, u ovom radu je za razvoj hardverskog odnosno softverskog dela projekta korišćen EDK alat. Ovaj alat značajno olakšava razvoj *embedded* sistema, jer je veliki deo posla automatizovan. Proces dizajniranja sistema se u dobroj meri svodi na serije odgovarajućih selekcija među ponuđenim opcijama, što je detaljno objašnjeno u radu. Takođe, EDK prilikom razvoja projekta generiše templejt fajlove sa preciznim uputstvima, tako da je i dodavanje novih funkcionalnosti značajno pojednostavljeno. U ovom radu je korišćena Virtex ML507 ploča sa MicroBlaze soft procesorom. Ovaj tip procesora ima veliki broj opcionih karakteristika koji korisnicima pruža veliku fleksibilnost u radu i mogućnost balansiranja između zahteva embedded sistema i zauzetih resursa FPGA čipa. Prednost ovakvog rešenja je jednostavnost, mogućnost lakog modifikovanja postojećih i dodavanja novih funkcionalnosti ili periferija. Takođe, u ovakvim sistemima nema potrebe za izmenama u hardveru (sem ako se sama struktura periferije menja), već se sve modifikacije vrše u softveru. Prednost se ogleda i u tome što se za različita procesiranja paketa može koristiti ista hardverska arhitektura bez zauzimanja dodatnih resursa, već se samo modifikuje i unapređuje softver. Mana ovakvog rešenja jeste manja brzina u odnosu na čisto hardversku implementaciju. Cilj ovog rada je da se predstave mogućnosti koje EDK alat nudi u razvoju embedded sistema. U tezi su data detaljna uputstva sa odgovarajućim slikama, tako da novi korisnici mogu odmah da počnu da koriste ove alate za svoje potrebe. U ovom radu je obrađena samo jedna od funkcija paketskog procesiranja koja se obavlja u ruterima, i ona predstavlja osnovu koja se lako može proširiti i drugim funkcijama paketskog procesiranja.

## LITERATURA

- [1] Zoran Čiča, Materijali sa predavanja iz predmeta Komutacioni sistemi
- [2] Z. Čiča, "Analysis and Implementation of Packet Processing Functions in Internet Routers," *Proc. of TELFOR 2012*, Belgrade, Serbia, November 2012.
- [3] MicroBlaze procesor reference guide, http://www.xilinx.com/support/documentation/sw\_manuals/mb\_ref\_guide.pdf
- [4] Rod Jesman, Fernando Martinez Vallina, Jafar Saniie, MicroBlaze Tutorial, http://ecasp.ece.iit.edu/tutorials/microblaze\_tutorial.pdf
- [5] EDK Concepts, Tools and Techniques, http://www.xilinx.com/support/documentation/sw\_manuals/xilinx14\_7/edk\_ctt.pdf