Virtualno sklopovlje za FPGA razvojni sustav

Sedlar, Stjepan

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:770555

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

VIRTUALNO SKLOPOVLJE ZA FPGA RAZVOJNI SUSTAV

Završni rad

Stjepan Sedlar

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 16.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Stjepan Sedlar
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4132, 28.07.2017.
OIB studenta:	44102817015
Mentor:	Izv. prof. dr. sc. Tomislav Matić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Virtualno sklopovlje za FPGA razvojni sustav
Znanstvena grana rada:	**Arhitektura računalnih sustava (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	16.09.2021.
Datum potvrde ocjene Odbora:	22.09.2021.

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

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2021.

Ime i prezime studenta:	Stjepan Sedlar
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4132, 28.07.2017.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Virtualno sklopovlje za FPGA razvojni sustav**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Tomislav Matić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD **1**

1.1. Zadatak završnog rada **1**

2. VIRTUALNO SKLOPOVLJE ZA UDALJENO UPRAVLJANJE FPGA RAZVOJNIM SUSTAVOM **2**

3. RAZVIJANJE VIRTUALNOG FPGA SKLOPOVLJA POMOĆU ARDUINO PLATFORME **4**

3.1. Sklopovlje **4**

3.1.1. Arduino 4

3.1.2. Arduino Pro Micro 4

3.1.3. Nexys 3 PMOD portovi 5

3.1.4. Shema spajanja 6

3.2. Portovi i registri Arduino mikroupravljača **7**

3.2.1. Rad s portovima i registrima u kôdu 7

3.3. Programska podrška za Arduino Pro Micro **8**

3.3.1. Arduino programsko okruženje 8

3.3.2. Logika i dijagram toka programa 9

3.3.3. Provjera konfiguracije i konfiguriranje 11

3.3.4. Ispis stanja s ulaznih nožica na serijski monitor 13

3.4. Programska podrška za FPGA razvojni sustav **14**

3.4.1. Xilinx ISE 14

3.4.2. VHDL kôd 14

4. TESTIRANJE **16**

5. ZAKLJUČAK **20**

LITERATURA **21**

SAŽETAK **22**

ABSTRACT **23**

ŽIVOTOPIS **24**

PRILOZI **25**

1. UVOD

Glavni cilj ovog završnog rada je razviti virtualno sklopovlje koristeći Arduino mikroupravljač, koje bi omogućilo upravljanje FPGA (engl. *Field Programmable Gate Array*) sustavom na daljinu. Motiv za ovaj rad je prvenstveno taj da se studentima, ali i zaposlenicima fakulteta omogući praktičan rad na daljinu. Time bi se moglo kombinirati fizičke laboratorije s FPGA sustavima i rad pomoću virtualnog sklopovlja te bi se znatno rasteretilo laboratorije. Virtualno sklopovlje također rješava još jedan od problema, a to je nepristupačna cijena FPGA razvojnih sustava. Studenti se tako uglavnom susreću s FPGA sustavima na laboratorijskim vježbama i nisu se u mogućnosti praktično pripremati za ispite. Uz pomoć virtualnog sklopovlja bi bilo omogućeno svim studentima da vrše razna testiranja po potrebi, van termina laboratorijskih vježbi, što znatno olakšava razumijevanje i usvajanje gradiva te omogućuje lakše povezivanje teoretskog i praktičnog znanja.

Ostatak rada organiziran je kako slijedi: u drugom poglavlju su ukratko opisani već postojeći slični radovi, poglavlje nakon toga obuhvaća sve najbitnije dijelove rada (sklopovlje i programsku podršku), u predzadnjem poglavlju je opisano testiranje, a zadnje poglavlje zaključuje rad.

1.1. Zadatak završnog rada

U radu je potrebno razviti i testirati sklopovlje koje će omogućiti upravljanje svijetlećim diodama i sklopkama FPGA razvojnog sustava pomoću računala bez potrebe fizičkog kontakta s razvojnim sustavom.

2. VIRTUALNO SKLOPOVLJE ZA UDALJENO UPRAVLJANJE FPGA RAZVOJNIM SUSTAVOM

U ovom poglavlju su ukratko opisani neki od znanstvenih radova iz istog područja.

U znanstvenom radu [1] se za izgled sučelja koristi eLML (engl. *eLesson Markup Language*) jezik za označavanje, koji je radni okvir (engl. *framework*) otvorenog kôda baziran na XML (*engl. Extensible Markup Language*) jeziku. Moodle web poslužitelj otvorenog kôda se koristi za pokretanje virtualnog sučelja, korisnici se putem istoga spajaju koristeći korisničko ime i lozinku. Također se koristi poslužitelj koji upravlja resursima. Poslužitelj za upravljanje resursima se nalazi u fizičkom laboratoriju te se na njega spaja potreban broj stvarnih FPGA sustava. FPGA se putem dodatne pločice sa konektorima povezuje s ulazno/izlaznom pločicom koja ima ugrađeno Ethernet sučelje i sposobnost pokretanja malog web poslužitelja. FPGA i ulazno/izlazna pločica se napajaju putem računala poslužitelja za upravljanje resursima. Korisnicima ovog virtualnog sklopovlja je također omogućen video prijenos fizičkog FPGA sustava u stvarnom vremenu.

U radu [2] se koristi USB mikroupravljač upravljan od strane web aplikacije koji se povezuje s računalom poslužiteljem putem USB porta, mikropuravljač se također povezuje s FPGA sustavom putem ugrađenog USB porta. USB sučelje pruža vezu između web aplikacije i povezanog mikroupravljača. Web aplikacija dodjeljuje svakom korisniku određeni FPGA sustav, njegovu pripadajuću kameru i USB mikroupravljač, također je odgovorna za stvaranje baze podataka i upravljanje istom. Baza podataka sadrži informacije o svakom korisniku, o njegovom programskom kôdu i korištenim FPGA sustavima. Nakon što se korisnik spoji na poslužitelj putem web aplikacije i postavi datoteku s programskim kôdom na poslužitelj, mikroupravljač konfigurira FPGA sustav. Web aplikacija nakon konfiguracije zahtjeva stanja FPGA sustava putem USB sučelja te koristeći HTTP (engl. *Hypertext Transfer Protocol*) protokol prenosi video FPGA sustava u stvarnom vremenu. Promjene stanja se također reflektiraju i prikazuju na web aplikaciji.

U radu [3] se koristi Windows aplikacija koja u pozadini radi sa poslužiteljem, tj. računalom u laboratoriju. Prilikom prvog otvaranja će se pojaviti korisničko sučelje s prikazom FPGA pločice i dodatna forma za prijavu korisnika. Nakon toga se putem aplikacije na poslužitelj podiže datoteka s programskim kôdom za FPGA sustav. Aplikacija detaljno ispisuje ako je došlo do pogreške ili ako je datoteka uspješno postavljena na server. Ukoliko je sve u redu, datoteka se preuzima s poslužitelja i „prebacuje“ na FPGA razvojni sustav. FPGA tada šalje stanja sklopki, LED dioda i ostalih elemenata natrag na poslužitelj putem mikroupravljača. Sve promjene se prikazuju na aplikaciji s FPGA sučeljem. Poslužitelj također upravlja pristupom virtualnom sklopovlju te

raspoređuje korisnike po prioritetima. Autori ovog rada ne koriste kameru za video prijenos u stvarnom vremenu.

U radu [4] autor ukratko opisuje model virtualnog laboratorija koji se sastoji od dva mikroupravljača i aplikacije s grafičkim sučeljem putem koje studenti upravljaju elementima FPGA sustava. Jedan mikroupravljač služi samo za pohranu datoteka s programskim kôdom i to koristeći FIFO (engl. *First In First Out*) model, a drugi mikroupravljač se koristi za dohvaćanje podataka, tj. programskog kôda s prvog mikroupravljača te njegovo „prebacivanje“ na FPGA i slanje povratnih informacija na računalo poslužitelj. Oba mikroupravljača i FPGA su spojeni na računalo. Za povratnu informaciju FPGA sustava se koristi samo video prijenos u stvarnom vremenu uz pomoć kamere.

U radu [5] nema dodatne mikroupravljačke pločice, već sve zadatke obavlja poslužitelj s Linux operacijskim sustavom. Na strani poslužitelja se također vodi briga o kontroli pristupa te je u pozadini pokrenuto nekoliko aplikacija i pogonskih programa napisanih u C jeziku koji su potrebni za komunikaciju poslužitelja direktno s FPGA sustavom. Na klijentskoj strani je potrebno samo instalirati preglednik te se spojiti na web aplikaciju koja prikazuje jednostavno sučelje s elementima FPGA sustava i dijelom za konfiguriranje FPGA sustava, tj. podizanje datoteke na poslužitelj. Ovaj rad ne koristi kameru za video prijenos.

3. RAZVIJANJE VIRTUALNOG FPGA SKLOPOVLJA POMOĆU ARDUINO PLATFORME

3.1. Sklopovlje

3.1.1. Arduino

Arduino je platforma otvorenog kôda (engl. *open source*) koja obuhvaća sklopovlje i programsku podršku, a ima različite primjene; od onih jednostavnih za edukaciju i testiranje pa sve do kompleksnijih u upravljačkim sustavima i robotici. Zbog svoje jednostavnosti i fleksibilnosti, platforma je pogodna za početnike, ali opet ostavlja naprednijim korisnicima dosta slobode. Programski kôd za Arduino platformu se piše u alatu Arduino IDE (engl. *Integrated Development Enviroment* – Integrirano Razvojno Okruženje) koji je opisan u dijelu 3.3.1. [6]

3.1.2. Arduino Pro Micro

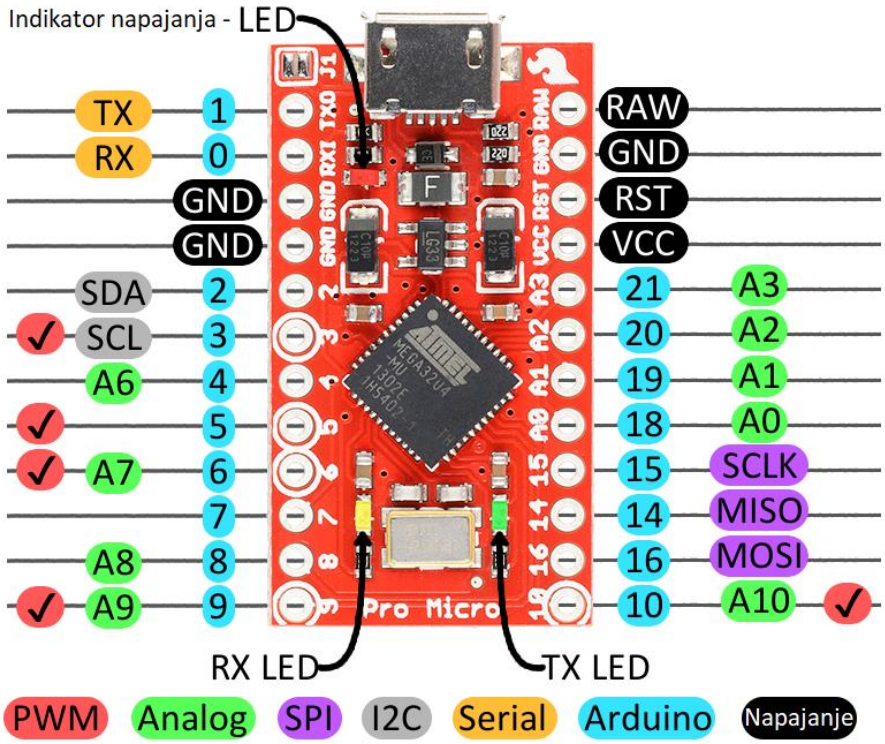
Za izradu ovoga rada je korištena pločica Arduino Pro Micro. Glavni dio pločice je ATmega32U4 mikroupravljač čije su specifikacije prikazane u tablici 3.1. Postoje dvije inačice ove pločice, jedna je s frekvencijom takta od 8 MHz i radnim naponom od 3.3 V (ova inačica je korištena u radu), a druga je s frekvencijom od 16 MHz i radnim naponom od 5 V. Pločica se napaja i komunicira s računalom putem micro USB-B porta. [7]

Tablica 3.1. Specifikacije mikroupravljača ATmega32U4.

Mikroupravljač	**ATmega32U4**
Frekvencija takta	8 MHz
Radni napon	3.3 V
Flash memorija	32 kB
EEPROM	1 kB
SRAM	2.5 kB

Pločica ukupno sadrži dvadeset i četiri izvedene nožice. Osamnaest nožica je programibilno. Sve nožice imaju sposobnost biti digitalni ulazi ili izlazi koji se mogu postaviti na logičku jedinicu ili nulu. Devet nožica mogu biti analogni ulazi te se kod njih koristi 10-bitni ADC (engl. *Analog-To-Digital Converter* – Analogno-Digitalni Pretvarač). Pet nožica ima PWM (engl. *Pulse-Width Modulation* – Modulacija Širine Impulsa) mogućnost, što znači da se korištenjem ovih nožica

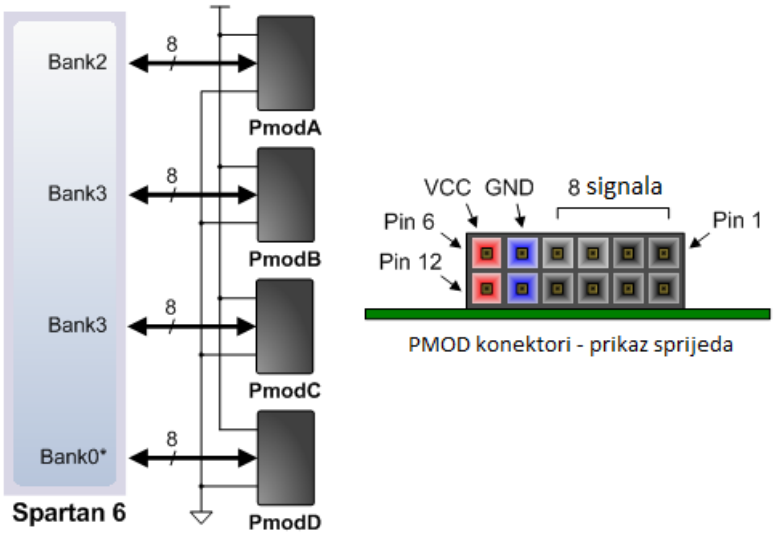
dobiva „privid“ analognih izlaza. Za rad je korišteno svih osamnaest programibilnih nožica. Na slici 3.1 se nalazi Pro Micro pločica s nožicama i njihovim oznakama.



Slika 3.1. Arduino Pro Micro.

3.1.3. Nexys 3 PMOD portovi

Testiranje je rađeno na Nexys 3 FPGA razvojnom sustavu. Izbor FPGA razvojnog sustava nije toliko bitan pa iz tog razloga on neće biti opisan kao Arduino mikroupravljač. Ukratko su opisani samo PMOD (engl. *Peripheral Module*) portovi.

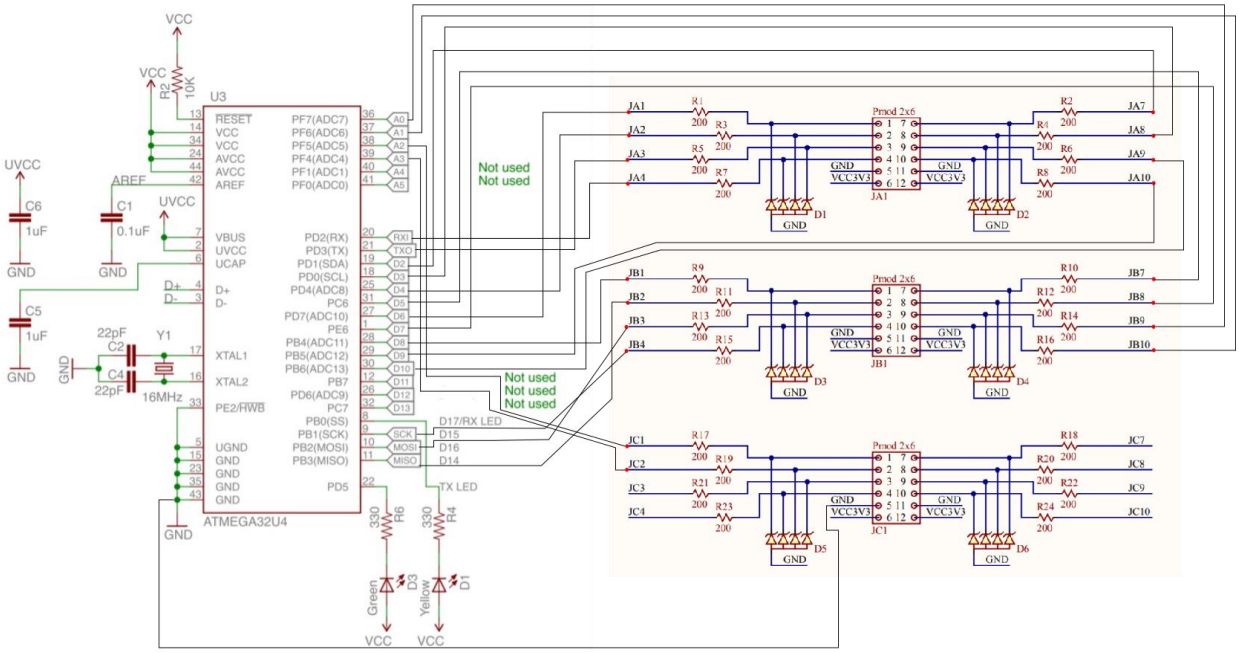


Slika 3.2. Nexys 3 PMOD portovi.

Nexys 3 sadrži četiri PMOD porta koji sadrže dva reda po šest kvadratnih 2.54 milimetarskih ženskih konektora. Svaki redak ima po jedan konektor za VCC signal od 3.3 V i po jedan za uzemljenje (slika 3.2). [8]

3.1.4. Shema spajanja

Na slici 3.3 lijevo je prikazan ranije spomenuti ATmega32U4 mikroupravljač, a desno su izdvojeni korišteni PMOD portovi s Nexys 3 FPGA razvojnog sustava. Njihove sheme su iz službenih dokumenata sa specifikacijama ([7], [8]), a spojene su u jednu pomoću programa „Inkscape“.



Slika 3.3. Električna shema spajanja.

3.2. Portovi i registri Arduino mikroupravljača

Arduino Pro Micro ima pet ulazno/izlaznih portova, svaki od tih portova je upravljan s tri registra. DDR (engl. *Data Direction Register*) registri upravljaju smjerom pojedinih nožica, svaki bit DDR registra je povezan s određenom nožicom. Ako je bit postavljen na 1, tada će odgovarajuća nožica biti postavljena kao izlaz, a ukoliko je bit postavljen na 0, tada će se ta nožica promatrati kao ulaz. PORT registri su podatkovni registri koji kontroliraju hoće li nožice biti postavljene na logičku nulu ili jedinicu. PIN registri su samo za čitanje i sadrže stanja svih ulaznih nožica, bitovi koji iznose 1 znače da su te nožice trenutno u logičkoj jedinici, a 0 znači da su u logičkoj nuli. [9]

Tablica 3.2. Arduino Pro Micro – portovi i registri.

PORT	**REGISTRI**	**Bit registra – nožica (najznačajniji bit lijevo)**							
D	DDRD, PORTD, PIND	6	-	-	4	1(TX)	0(RX)	2	3
B	DDRB, PORTB, PINB	-	10	9	8	14	16	15	-
C	DDRC, PORTC, PINC	-	5	-	-	-	-	-	-
E	DDRE, PORTE, PINE	-	7	-	-	-	-	-	-
F	DDRF, PORTF, PINF	A0	A1	A2	A3	-	-	-	-

Kao što je vidljivo iz tablice 3.2, registri su 8-bitni, ali niti jedan registar nije u potpunosti iskorišten, također nožice nisu „po redu“ raspoređene po bitovima registara. Cijelo programsko rješenje zbog toga ispada kompleksno.

3.2.1. Rad s portovima i registrima u kôdu

Ugrađene Arduino funkcije za očitavanje stanja pojedinih nožica i njihovo postavljanje nisu optimizirane za brzinu izvođenja, već su prilagođene tako da budu intuitivnije, da vizualno izgledaju ljepše u kôdu i da budu jednostavnije za korištenje. Pošto se u radu koristi veliki broj nožica, pri promjeni njihovih stanja na većim frekvencijama ove funkcije stvaraju problem i ne rade ispravno. Iz ovog razloga se portovi i registri mikroupravljača koriste direktno u kôdu. Svi registri i portovi su unaprijed definirani kao varijable u Arduinu. [10]

Tako primjerice linije kôda sa slika 3.4 i 3.5 rade istu stvar na različit način. Veliko slovo B ispred niza binarnih znamenki znači da će se broj interpretirati kao binarni, tj. moguće je pristupiti svakom bitu tog broja te ga promijeniti.

Linija ***Kôd***

```
1:      pinMode(6, OUTPUT);
2:      pinMode(4, OUTPUT);
3:      digitalWrite(6, HIGH);
4:      digitalWrite(4, LOW)
```

Slika 3.4. Korištenje ugrađenih Arduino funkcija.

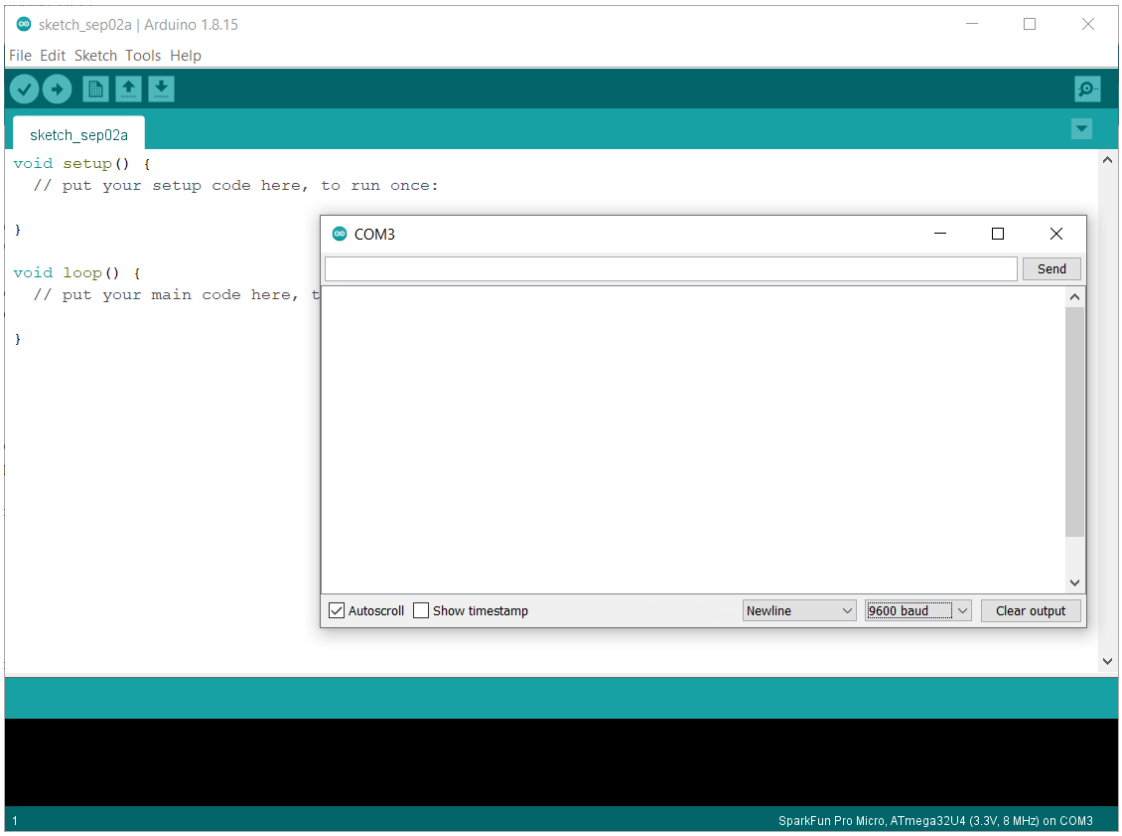
Linija ***Kôd***

```
1:      DDRD = B10010000;
2:      PORTD = B10000000;
```

Slika 3.5. Korištenje registara.

3.3. Programska podrška za Arduino Pro Micro

3.3.1. Arduino programsko okruženje



Slika 3.6. Arduino IDE korisničko sučelje s otvorenim serijskim monitorom.

Arduino IDE je programski alat koji se koristi za pisanje i kompiliranje programskog kôda kao i

za njegovo učitavanje na Arduino pločicu. Sučelje ovog programa je vrlo intuitivno i jednostavno za korištenje, pogotovo u usporedbi s ostalim okruženjima (slika 3.6). [11]

Arduino okruženje automatski generira osnovni dio Arduino programa kako bi on mogao funkcionirati, funkcije „setup()“ i „loop()“. Na alatnoj traci za brzi pristup se nalazi šest opcija.

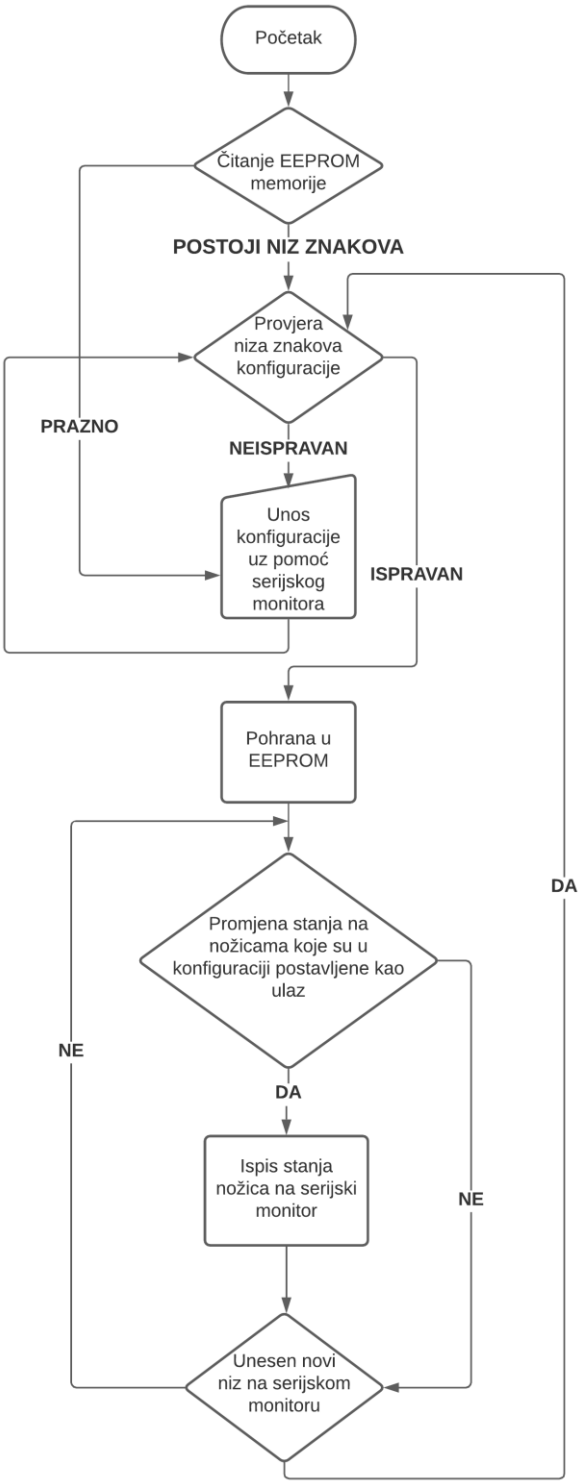
S lijeva na desno:

- *Verify*
- *Upload*
- *New*
- *Open*
- *Save*
- *Serial Monitor*.

Najbitnije i najviše korištene opcije su *Upload* i *Serial Monitor*, prva opcija prebacuje programski kôd (*sketch*) na Arduino pločicu, a druga otvara ugrađeni Arduinov alat za serijsku komunikaciju *Serial Monitor*. Pri otvaranju serijskog monitora se prikazuje novi prozor u kojemu se putem tipkovnice šalju podaci na Arduino te se prikazuju podaci koji prema programskom kôdu trebaju biti ispisani na serijski monitor.

3.3.2. Logika i dijagram toka programa

Radi jednostavnosti i lakše analize, kôd je podijeljen na najvažnije dijelove. Kompletan kôd, je dostupan u prilozima, no kako bi bilo lakše pratiti logiku, napravljen je prikaz dijagrama toka cijelog programa.



Slika 3.7. Dijagram toka Arduino programa.

Program se temelji na radu sa znakovima i *String* objektima radi jednostavnosti i već ugrađenih korisnih metoda. Na samom početku se čita iz EEPROM (engl. *Electrically Erasable Programmable Read-Only Memory*) memorije kako bi se provjerilo ako postoji spremljen ispravan niz znakova koji predstavlja konfiguraciju. Ukoliko ne postoji tada je obavezno unijeti

konfiguracijski niz znakova putem serijskog monitora. Kada je konfiguracija ispravna, moguće ju je ponovno unijeti ili je moguće samo unijeti željena stanja za nožice koje su u konfiguraciji postavljene kao izlaz. Program također konstantno provjerava ukoliko je došlo do promjene stanja na nožicama koje su postavljene kao ulaz te potom ispisuje ta stanja (slika 3.7).

3.3.3. Provjera konfiguracije i konfiguriranje

Provjera konfiguracije se vrši pomoću funkcije „checkConfig()“ koja je prikazana na slici 3.8.

Linija ***Kôd***

```
1:      bool checkConfig(String conf)
2:      {
3:        bool isCorrect = false;
4:        for (int i = 0; i < conf.length(); i++)
5:        {
            if (conf[i] == '#' || conf[i] == 'O' || conf[i] == 'I' || conf[i]
        == '1' || conf[i] == '0' || conf[i] == 'o' || conf[i] == 'i' ||
6:      conf[i] == '\n')
7:            isCorrect = true;
8:          else
9:            return false;
10:       }
11:       return isCorrect;
12:     }
```

Slika 3.8. Kôd funkcije za provjeru konfiguracije.

Funkcija kao parametar prima *String* objekt te prolazi kroz svaki znak pojedinačno i provjerava ga. Ukoliko jedan od znakova nije pravilan znak za konfiguraciju ('i', 'I', 'o', 'O', 1, 0), automatski se prekida izvođenje te funkcija vraća 0 odnosno *false*. Na slici 3.9 su dani primjeri ispravnog niza znakova.

```
1:      „#IIIIIIIIIIIIIIIII#“
2:      „#IIIIOOOOIIIIOOOOII#11001100“
3:      „#OOOOOOOOOOOOOOOOOO#111111110000001111“
```

Slika 3.9. Primjeri ispravnog konfiguracijskog niza znakova.

Konfiguracijski niz počinje sa znakom ljestvi ('#') te nakon toga slijedi osamnaest znakova kod kojih 'I' predstavlja ulaz, a 'O' izlaz, što znači da će se nožice po redu tako postavljati. Za svaku nožicu koja je postavljena kao izlaz, nakon znaka ljestvi mora biti jedan znak 0 ili 1 koji postavlja izlaz na logičku nulu ili logičku jedinicu.

Nakon provjere ispravnosti konfiguracije dolazimo jednog od glavnih dijelova programa, funkcije „configSetup()“ (slika 3.10). Cijeli kôd funkcije nije postavljen unutar samog rada zbog složenosti i zbog velikog broja linija.

Linija ***Kôd***

```
1:      void configSetup();
```

Slika 3.10. Deklaracija funkcije „configSetup()“.

Ova funkcija ne prima parametre, već radi s globalnim varijablama i registrima Arduino mikroupravljača. Zapisuje konfiguracijski niz u EEPROM te ga nakon toga dijeli u više nizova znakova od kojih svaki predstavlja niz za jedan registar, tj. port. Fizički registri i portovi mikroupravljača su ranije objašnjeni u potpoglavlju 3.2. Svaki od nizova znakova se pretvara u *byte* tip podatka kako bi se omogućilo postavljanje portova na vrijednosti 0 ili 1.

Pretvorbe se rade uz pomoć funkcija „IOStringToByte()“ i „IOStringToPortByte()“. Funkcije rade na sličan način, pa će biti objašnjena samo jedna i to „IOStringToPortByte()“ (slika 3.11).

Linija ***Kôd***

```
1:      byte IOStringToPortByte(byte portValue, String IO){
2:        byte result = 0;
3:        int counter = 0;
4:        for (int i = 0; i < 8; i++)
5:        {
6:          if (bitRead(portValue, 8 - i - 1))
7:          {
8:            if (IO[counter] == '1')
9:            {
10:             bitWrite(result, 8 - i - 1, 1);
11:             counter++;
12:           }
13:           else
14:           {
15:             bitWrite(result, 8 - i - 1, 0);
16:             counter++;
17:           }
18:         }
19:         else
20:           bitWrite(result, 8 - i - 1, 0);
21:       }
22:       return result; }
```

Slika 3.11. Kôd funkcije „IOStringToPortByte()“.

Funkcija kao parametre prima *byte* tip podatka koji će predstavljati DDR registar koji određuje koja je nožica ulaz, a koja izlaz i niz znakova koji predstavlja vrijednosti nožica koje su postavljene kao izlaz. Petlja prolazi kroz svaki od osam bitova predanog *byte* argumenta te ih čita uz pomoć ugrađene funkcije „bitRead()“. Ukoliko je neki bit postavljen na 1, provjerava se odgovarajući znak u predanom nizu te se vrijednost znaka zapisuje na bit lokalno stvorene *byte* varijable koju će funkcija vratiti kao rezultat i uveća se brojač za pomicanje kroz niz znakova. Brojač je potreban pošto duljina predanog niza znakova ne mora biti osam kao broj bitova u *byte* tipu podatka. Na ovaj način se istovremeno prolazi kroz niz znakova i svaki od bitova DDR registra Ako je pak bit postavljen na 0 to znači da je ta nožica ulazna te se na odgovarajući bit rezultata tada također zapisuje 0. Rezultat funkcije predstavlja vrijednost odgovarajućeg PORT registra te se kasnije koristi za njegovo postavljanje.

Funkcija „IOStringToByte()“ prima samo niz znakova kao argument, stvara lokalnu *byte* varijablu i provjerava predani niz. Ako je znak u nizu 'o' ili 'O', u odgovarajući bit *byte* varijable se zapisuje 1, a u suprotnom 0. ('I' – engl. *Input* – ulaz – 0, 'O' – engl. *Output* – izlaz – 1).

3.3.4. Ispis stanja s ulaznih nožica na serijski monitor

Drugi najvažniji dio kôda je funkcija za pretvorbu PIN registara u nizove znakova za ispis na serijskom monitoru i provjeru ukoliko je došlo do promjene stanja na nekom bitu tih registara. Ova funkcija također zbog velikog broja linija neće biti prikazana cijela, već samo njena deklaracija (slika 3.12).

Linija ***Kôd***

```
1:      String PINregistersToString();
```

Slika 3.12. Deklaracija funkcije „PINRegistersToString()“.

Funkcija također ne prima parametre već radi s globalnim varijablama. Čita se svaki od DDR registara po bitovima uz pomoć već spomenute ugrađene funkcije „bitRead()“, ako je bit postavljen kao ulaz (0), tada se čita pripadajući bit iz odgovarajućeg PIN registra. Očitani bit iz PIN registra se pohranjuje kao znak u lokalno stvoren niz znakova za svaki od PIN registara. Nizovi znakova se na kraju funkcije međusobno spajaju i funkcija vraća jedan niz znakova koji je praktičan za ispisivanje na serijskom monitoru, a on predstavlja vrijednosti svih ulaznih nožica mikroupravljača.

3.4. Programska podrška za FPGA razvojni sustav

3.4.1. Xilinx ISE

Xilinx ISE (engl. *Integrated Synthesis Environment*) je programski alat koji se koristi za sintezu, implementaciju i analizu HDL (engl, *Hardware Description Language*) kôda. Pruža mogućnosti podešavanja svih potrebnih postavki nakon čega se HDL kôd „prebacuje“ na FPGA razvojni sustav. [12]

3.4.2. VHDL kôd

VHDL (engl. *Very High Speed Integrated Circuit Hardware Description Language*) kôd nije primarni zadatak ovoga rada pa neće biti objašnjen kao Arduino kôd. Ovaj dio ovisi o korisniku razvijenog virtualnog sklopovlja te je potreban samo kako bi se testirala ispravnost konačnog rada. [13]

U Xilinx okruženju je potrebno postaviti željene pinove, LED-ice ili sklopke kao ulaze ili kao izlaze, za što je prvo potrebno konfigurirati Arduino kako je već objašnjeno. Ulazi FPGA razvojnog sustava su izlazi na Arduinu i obrnuto, u ovome dijelu će se pod ulazima smatrati ulazi FPGA razvojnog sustava, a izlazima njegovi izlazi. Ulazi primaju informacije, tj. logičke nule i jedinice preko PMOD portova na koje su spojene nožice Arduino pločice, dok izlazi šalju informacije, također preko PMOD portova.

VHDL kôd provjerava ponašanje Arduino mikroupravljača kada FPGA razvojni sustav šalje informacije pri različitim frekvencijama kako bi se provjerila ograničenja mirkoupravljača te također provjerava ponašanje FPGA sustava kada se informacije šalju s mikroupravljača. Na slici 3.13 je prikazan „process“ dio kôda koji se koristi za testiranje kada su sve nožice Arduino mikroupravljača korištene kao ulazi.

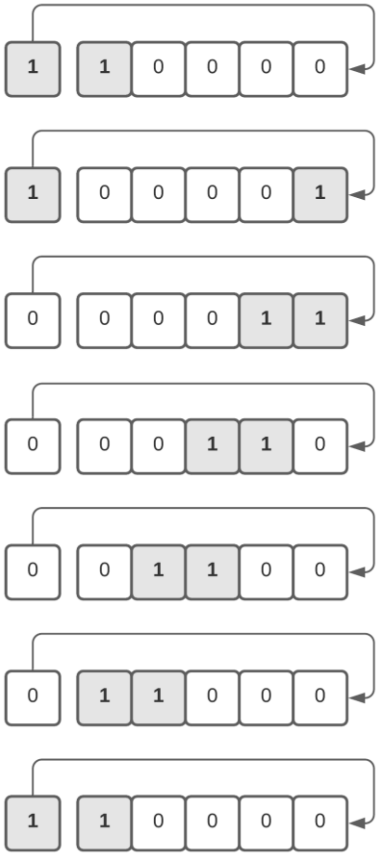
Linija ***Kôd***

```
1:      process (clk_out, rst)
2:      begin
3:        if(rst='1')then
4:          pmod(nBits-1 downto 0) <= "11" & (nBits-3 downto 0 => '0');
5:        elsif(clk_out'event and clk_out = '1') then
6:          pmod <= pmod(nBits-2 downto 0) & pmod(nBits-1);
7:        end if;
8:      end process;
```

Slika 3.13. Dio VHDL kôda za pomicanje bitova u lijevo.

Varijabla „nBits“ predstavlja željeni broj izlaza FPGA sustava, što je u ovom slučaju osamnaest kako je i postavljeno u dijelu kôda koji nije prikazan. Proces je osjetljiv na varijable „rst“ koja je dodijeljena jednoj od tipki na FPGA sustavu i „clk_out“ koja predstavlja izlaznu frekvenciju djelitelja frekvencije. Ukoliko se pritisne tipka za reset, dva najznačajnija bita varijable „pmod“ se postavljaju na 1, a ostali na 0 i tako ostaje sve dok je tipka pritisnuta. Ako tipka nije pritisnuta, za svaki rastući brid varijable „clk_out“ se vrši pomicanje varijable „pmod“ u lijevo za jedan bit (engl. *Bit Shifting*).

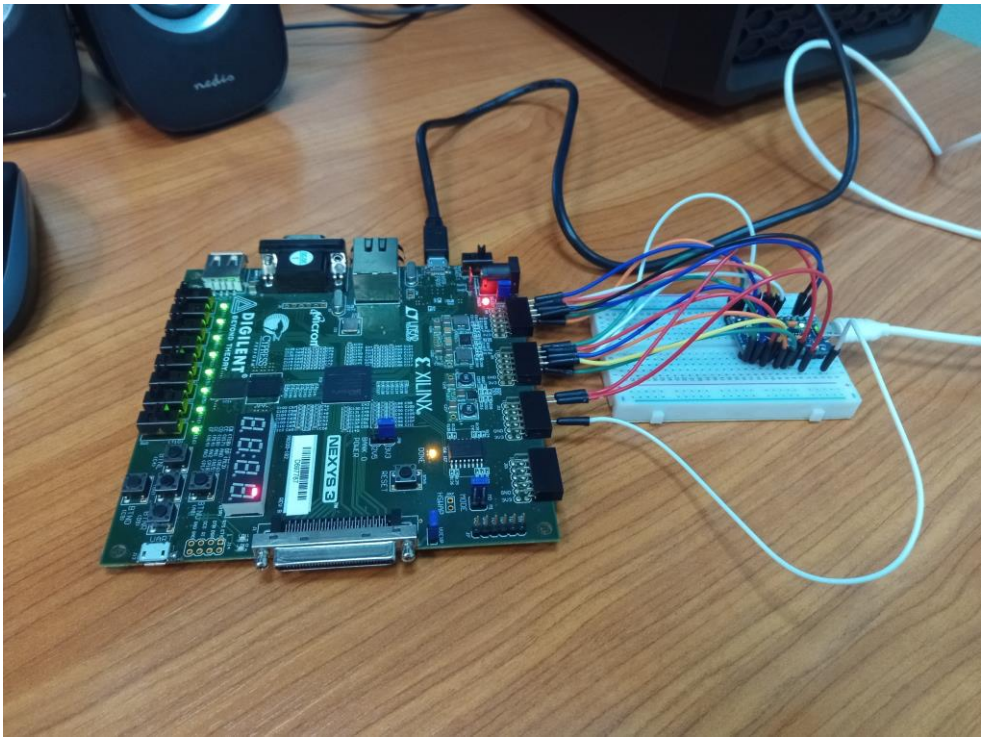
Na slici 3.14 je prikazano kako funkcionira pomicanje bitova u lijevo za jedan bit. U primjeru je korišteno šest FPGA izlaza („nBits“ = 6), no princip je isti i za više bitova.



Slika 3.14. Prikaz ispravnog pomicanja bitova u lijevo za jedan bit („nBits“ = 6).

4. TESTIRANJE

Nakon što je sve spojeno kao prema shemi, potrebno je konfigurirati prvo Pro Micro, a zatim i FPGA. To znači da je potrebno prebaciti programski kôd na Arduino pločicu te ga pokrenuti i unijeti konfiguracijski niz znakova. Kada je Pro Micro podešen, poznato je koje od njegovih nožica su ulazne, a koje izlazne što je potrebno kako bi bilo moguće postaviti smjer pojedinačnih podatkovnih konektora PMOD portova FPGA sustava. Na slici 4.1 je prikazan spoj FPGA i mikroupravljača za konfiguraciju kada je pola nožica mikroupravljača postavljeno kao ulaz, a pola kao izlaz te su izlazne postavljene u logičku jedinicu, što se vidi na LED diodama i na jednom segmentu pokaznika.



Slika 4.1. Spajanje Arduino mikroupravljača s FPGA sustavom.

Testiranje je odrađeno za slučajeve kada je dio nožica mikroupravljača postavljen kao ulaz, a dio kao izlaz, kada su sve nožice postavljene kao ulaz i sve kao izlaz. Testirano je ponašanje kada se vrijednosti izlaza na FPGA sustavu i Arduino mikroupravljaču mijenjaju ručno, tj. malom brzinom i kada se FPGA izlazi mijenjaju brzinom za nekoliko frekvencija u rasponu od 50 Hz do 3 kHz, kako bi se provjerila ograničenja brzine izvođenja samog kôda mikroupravljača i potencijalna ograničenja serijske komunikacije.

Kada se za FPGA izlaz koristi pola nožica mikroupravljača, on radi ispravno na 2.5 kHz. Na slici 4.2 je prikazan ispis stanja ulaznih nožica mikroupravljača za navedenu frekvenciju.

Linija	***Ispis na serijskom monitoru***
1:	18:27:09.408 -> 110000000
2:	18:27:09.408 -> 100000001
3:	18:27:09.408 -> 000000011
4:	18:27:09.408 -> 000000110
5:	18:27:09.408 -> 000001100
6:	18:27:09.408 -> 000011000
7:	18:27:09.408 -> 000110000
8:	18:27:09.408 -> 001100000
9:	18:27:09.408 -> 011000000
10:	18:27:09.408 -> 110000000
11:	18:27:09.408 -> 100000001
12:	18:27:09.408 -> 000000011
13:	18:27:09.408 -> 000000110
14:	18:27:09.408 -> 000001100
15:	18:27:09.408 -> 000011000
16:	18:27:09.408 -> 000110000
17:	18:27:09.408 -> 001100000
18:	18:27:09.408 -> 011000000
19:	18:27:09.408 -> 110000000

Slika 4.2. Ispis stanja ulaza mikroupravljača na serijskom monitoru za devet ulaza i konfiguracijski niz znakova: "#IIIIIIIIIOOOOOOOOO#111111111".

Na temelju ispisa možemo zaključiti da je rad mikroupravljača ispravan za 2.5 kHz. U VHDL kôdu se konstantno izvršava pomicanje za jedan bit u lijevo, a ispis pokazuje da niti jedan bit nije preskočen. Ako se frekvencija poveća na 3 kHz, tada dolazi do preskakanja nekih od bitova.

U slučaju kada se sve nožice mikroupravljača koriste kao FPGA izlaz, za 2 kHz dobivamo ispis sa slike 4.3.

Linija	***Ispis na serijskom monitoru***
1:	18:47:38.322 -> 110000000000000000
2:	18:47:38.322 -> 100000000000000001
3:	18:47:38.322 -> 000000000000000011
4:	18:47:38.322 -> 000000000000000110
5:	18:47:38.322 -> 000000000000001100
6:	18:47:38.322 -> 000000000000011000
7:	18:47:38.322 -> 000000000000110000
8:	18:47:38.322 -> 000000000001100000
9:	18:47:38.322 -> 000000000011000000
10:	18:47:38.322 -> 000000000110000000
11:	18:47:38.322 -> 000000001100000000
12:	18:47:38.322 -> 000000011000000000
13:	18:47:38.322 -> 000000110000000000
14:	18:47:38.322 -> 000001100000000000
15:	18:47:38.322 -> 000011000000000000
16:	18:47:38.322 -> 000110000000000000
17:	18:47:38.322 -> 001100000000000000
18:	18:47:38.322 -> 011000000000000000
19:	18:47:38.322 -> 110000000000000000
20:	18:47:38.322 -> 100000000000000001
21:	18:47:38.322 -> 000000000000000011
22:	18:47:38.322 -> 000000000000000110
23:	18:47:38.322 -> 000000000000001100
24:	18:47:38.322 -> 000000000000011000
25:	18:47:38.322 -> 000000000000110000
26:	18:47:38.322 -> 000000000001100000

Slika 4.3. Ispis stanja ulaza mikroupravljača na serijskom monitoru za devet ulaza i konfiguracijski niz znakova: "#IIIIIIIIIIIIIIII#".

Ovdje je također vidljivo da se ne preskače niti jedan bit te da je rad mikroupravljača za ovaj slučaj ispravan na 2 kHz. Ukoliko se frekvencija poveća na 2.5 kHz kao u prethodnom slučaju, dolazi do preskakanja bitova.

U tablici 4.1 je prikazano koji se PMOD port i konektor FPGA sustava povezuje na koju nožicu Arduino mikroupravljača. Tablica je rađena prema električnoj shemi sa slike 3.3.

Tablica 4.1. Povezivanje FPGA sustava i Arduino mikroupravljača.

PMOD port	Konektor : Oznaka	Pro Micro nožica
A	JA1 : T12	6
	JA2 : V12	4
	JA3 : N10	1(TX)
	JA4 : P11	0(RX)
	JA7 : M10	2
	JA8 : N9	3
	JA9 : U11	10
	JA10 : V11	9
B	JB1 : K2	8
	JB2 : K1	14
	JB3 : L4	16
	JB4 : L3	15
	JB7 : J3	5
	JB8 : J1	7
	JB9 : K3	A0
	JB10 : K5	A1
C	JC1 : H3	A2
	JC2 : L7	A3

5. ZAKLJUČAK

U ovom završnom radu je pomoću Arduino platforme razvijeno virtualno sklopovlje koje se može koristiti za upravljanje FPGA sustavom bez potrebe fizičkog kontakta sa sustavom. Kako bi se koristilo virtualno sklopovlje, potrebno je imati kontrolu samo nad Arduino mirkoupravljačem. Prilikom pisanja programskog kôda za mikroupravljač su se pojavljivale različite poteškoće i ograničenja, posebno zbog odabira *String* klase za rad sa znakovima. *String* klasa je korištena zbog korisnih ugrađenih funkcija, no pojavljuje se problem zbog sporog izvođenja kôda pri samom stvaranju objekata te klase iz razloga što se memorija zauzima dinamički. Problem se ne javlja pri slanju informacija s mikroupravljača na FPGA, već obratno. Zbog sporog izvođenja kôda, mikroupravljač nije ispravno prepoznavao i prikazivao promjene stanja svojih nožica i to već na malim frekvencijama. Ovaj problem je riješen na način da se kombinira korištenje *String* klase s klasičnim nizovima znakova te se time puno dobiva na brzini izvođenja. Mikroupravljač može ispravno prikazati promjene sve do 2.5 kHz za devet i 2 kHz za svih osamnaest ulaznih nožica, ali konačni rad ostavlja još puno prostora za napredak.

Moguće je svugdje koristiti klasične nizove znakova te na taj način još dobiti na brzini, također se može razviti Windows aplikacija koja bi odrađivala u pozadini svu serijsku komunikaciju umjesto da se direktno koristi serijski monitor. Moguće je razviti tiskanu pločicu da bude posrednik u komunikaciji između FPGA i mikroupravljača, na ovaj način bi se moglo galvanski odvojiti i zaštititi FPGA sustav i mikroupravljač što je uvijek preporučeno, to bi također i vizualno izgledalo bolje. Moguće je dodati kameru koja bi u stvarnom vremenu prikazivala promjene na FPGA sustavu te omogućiti udaljeno spajanje na računalo na koje su spojeni FPGA i mikroupravljač.

LITERATURA

[1] M., Reichenbach, M., Schmidt, B., Pfundt, D., Fey, „A New Virtual Hardware Laboratory for Remote FPGA Experiments on Real Hardware“, tra. 2012.

[2] F., Morgan i ostali, „Remote FPGA Lab with Interactive Control and Visualisation Interface“, str. 496–499, 2011.

[3] T. M., Hoang i ostali, „IMPLEMENTATION OF LOW COST FPGA REMOTE LABORATORY“, ASEAN Engineering Journal, izd. 1, sv. 5, str. 56–76, 2015.

[4] S., Das, „A Novel Approach to FPGA Remote Laboratory“, 2016.

[5] A. F., Herrero, I., Elguezábal, M. L., Vallejo, „A WEB-BASED ENVIRONMENT PROVIDING REMOTE ACCESS TO FPGA PLATFORMS FOR TEACHING DIGITAL HARDWARE DESIGN“, str. 5.

[6] „What is Arduino?“ [online]. Dostupno na: https://www.arduino.cc/en/Guide/Introduction. [Pristupljeno: 18.8.2021.].

[7] „Pro Micro - 3.3V/8MHz - DEV-12587 - SparkFun Electronics“ [online]. Dostupno na: https://www.sparkfun.com/products/12587. [Pristupljeno: 18.8.2021.].

[8] „Nexys 3 - Digilent Reference“ [online]. Dostupno na: https://digilent.com/reference/programmable-logic/nexys-3/start. [Pristupljeno: 14.9.2021.].

[9] „ATmega16U4/32U4 Datasheet“, str. 438.

[10] „Arduino - PortManipulation“ [online]. Dostupno na: https://www.arduino.cc/en/Reference/PortManipulation. [Pristupljeno: 1.9.2021.].

[11] „Arduino Software (IDE)“ [online]. Dostupno na: https://www.arduino.cc/en/Guide/Environment. [Pristupljeno: 10.9.2021.]

[12] „ISE Help“ [online]. Dostupno na: https://www.xilinx.com/html_docs/xilinx14_7/isehelp_start.htm. [Pristupljeno: 13.9.2021.].

[13] V. A., Pedroni, Circuit design and simulation with VHDL, 2nd ed. MIT Press: Cambridge, Mass, 2010.

SAŽETAK

Cilj ovog rada je bio uz pomoć Arduino platfome razviti virtualno sklopovlje koje bi se potencijalno moglo nadograditi i koristiti za udaljeno upravljanje FPGA razvojnim sustavom. Rad je realiziran uz pomoć Arduino Pro Micro pločice na način da se putem serijske komunikacije na Arduino pločicu šalje konfiguracijski niz znakova koji se potom pohranjuje u EEPROM mikroupravljača. U konfiguraciji se odabire koje izvedene nožice mikroupravljača će biti ulazne, a koje izlazne. Kada je ispravno konfigurirana, pločica se povezuje putem PMOD portova s FPGA sustavom gdje se prema konfiguraciji mikroupravljača postavljaju ulazi i izlazi FPGA razvojnog sustava.

Ključne riječi: virtualno sklopovlje, Arduino, mikroupravljač, FPGA, PMOD

ABSTRACT

Title: Virtual Hardware for FPGA Based Development Board

The aim of this thesis was to develop virtual hardware using the Arduino platform that could potentially be upgraded and used for the remote controlling FPGA development board. This is accomplished using the Arduino Pro Micro board in a way that a configuration character string is sent to the Arduino board via serial communication, which is then stored in the EEPROM of the microcontroller. The configuration is used to determine which derived pins will be inputs and which outputs. When properly configured, the board is then connected to FPGA via PMOD ports, where the inputs and the outputs of the FPGA development board are set based on the microcontroller configuration.

Keywords: virtual hardware, Arduino, microcontroller, FPGA, PMOD

ŽIVOTOPIS

Stjepan Sedlar rođen je 20.7.1998. u Novoj Gradišci. Odrastao je u Kutini gdje je pohađao Osnovnu školu Stjepana Kefelje koju je završio 2013. Iste godine upisuje smjer tehničar za računalstvo u Tehničkoj školi Kutina. Završava ju 2017. te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo, kao prvi odabir. Sada je na zadnjoj godini preddiplomskog sveučilišnog studija računarstvo te planira upisati diplomski studij na istome fakultetu.

_____

Potpis autora

PRILOZI

Na CD-u nalaze se sljedeći prilozi:

- Arduino programski kôd
- Xilinx ISE projekt s VHDL programskim kôdom
- Završni rad u .pdf formatu
- Završni rad u .docx formatu.