

Sveučilište u Zagrebu  
Fakultet elektrotehnike i računarstva

## **Digitalna logika**

### **Laboratorijske vježbe korištenjem sklopovskih pomagala**

#### **Upute za 6. laboratorijsku vježbu**

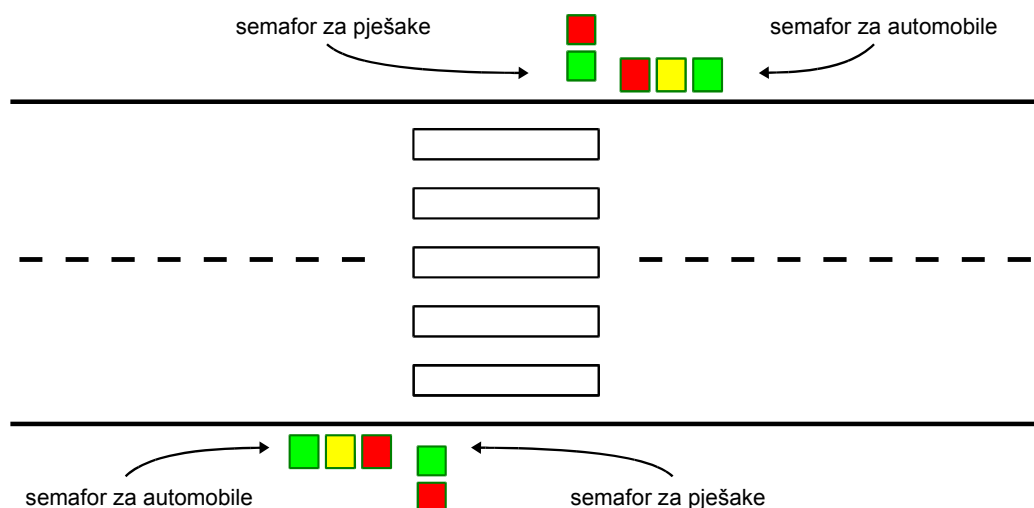
Marko Zec

Siječanj 2018.

# 1 Automat za upravljanje semaforom (2 boda)

## 1.1 Problem: pokvareni semafor

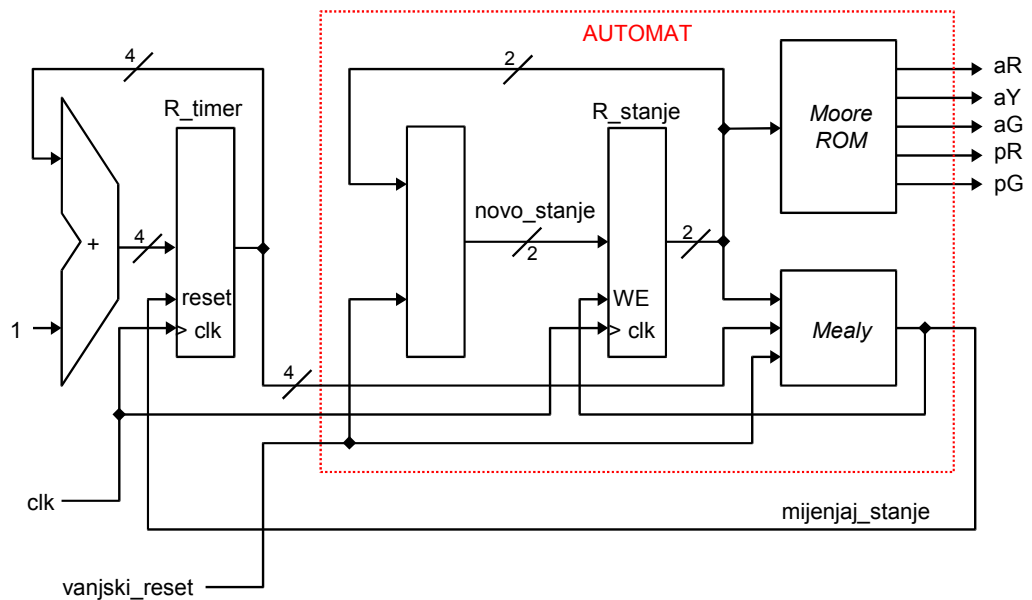
Vaš je zadatak već postojeći, nefunkcionalni prototip upravljačkog modula jednostavnog semafora uz što manje preinaka osposobiti za rad u skladu s funkcijskim specifikacijama.



Slika 1: jednostavni semafor za automobile i pješake

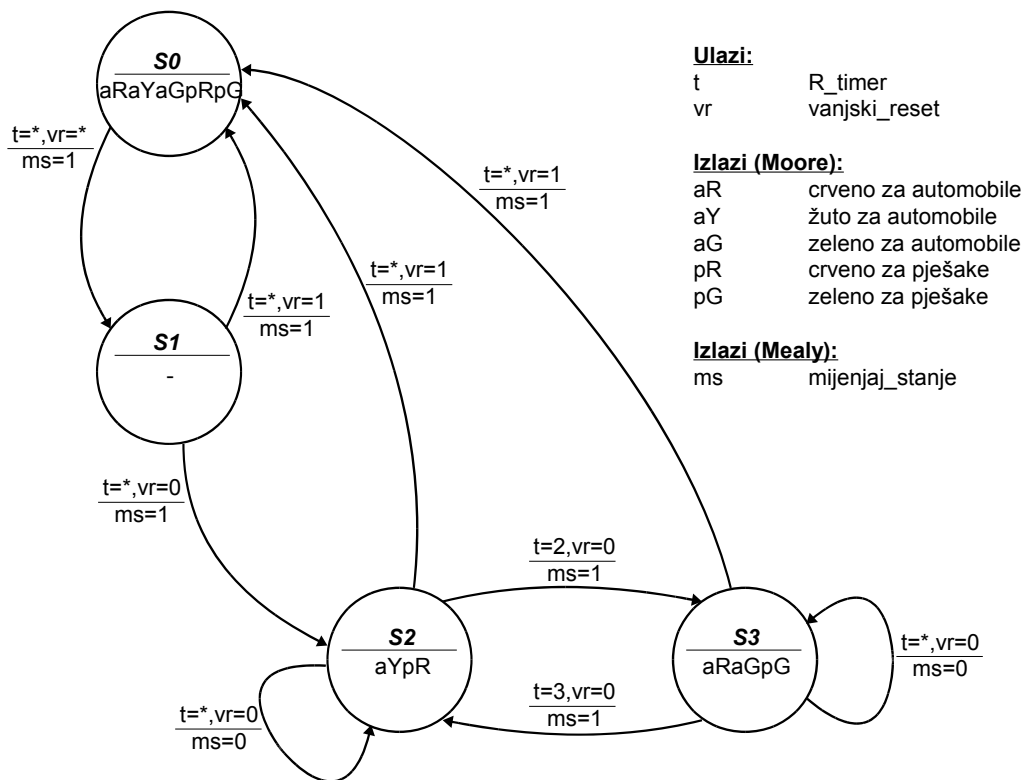
Prototip sklopa koji treba upravljati semaforom sa slike 1 zamišljen je kao automat s konačnim brojem stanja, koji zavisno od ulaza `vanjski_reset` treba raditi u jednom od dva moguća režima rada. Ukoliko je signal `vanjski_reset` postavljen na 1, na semaforu za automobile treba treptati žuto svjetlo, a semafor za pješake treba biti ugašen. Kad je signal `vanjski_reset` postavljen na 0, semafor treba beskonačno ponavljati uobičajeni ciklus paljenja i gašenja signalnih svjetla za automobile i pješake. Za postizanje različitih vremena zadržavanja u pojedinom stanju (npr. u stanju u kojem svijetli zeleno svjetlo automat se treba zadržati dulje nego u stanju u kojem svijetli žuto svjetlo) koristi se vanjski brojač vremenskih impulsa nazvan `timer`, a kojeg se može postaviti na nulu (resetirati) odgovarajućim izlaznim signalom iz automata. Postavljanjem ulaznog signala `vanjski_reset` na 1 automat se u prvom slijedećem ciklusu takta treba prebaciti u režim treptajućeg žutog svjetla bez obzira na trenutno stanje u kojem se nalazi. Rad semafora sinkroniziran je signalom takta frekvencije cca. 1.5 Hz.

Automat za upravljanje radom semafora sastoji se od memorijskog elementa (registra) kojim je određeno trenutno stanje, te kombinacijskih modula kojima se zavisno od ulaza i trenutnog stanja određuju izlazi iz automata, kao i eventualni prijelaz automata u novo stanje. Izlazni signali koji ovise isključivo o trenutnom stanju automata upravljani su kombinacijskim sklopom koji je izveden kao memorija za čitanje (ROM), na čiji se adresni ulaz dovodi kodna riječ trenutnog stanja, a izlazi direktno upravljaju signalnim indikatorima. Signal `mijenjaj_stanje` služi za postavljanje vanjskog brojača ciklusa takta na nulu kod prelaska na novo stanje, a ovisi i o trenutnom stanju i o ulazima u automat. Struktura modula za upravljanje semaforom koji se sastoji od automata i upravljivog brojača ciklusa takta prikazana je blok-shemom na slici 2.



Slika 2: struktura upravljačkog modula `sem_automat_pokvareni`

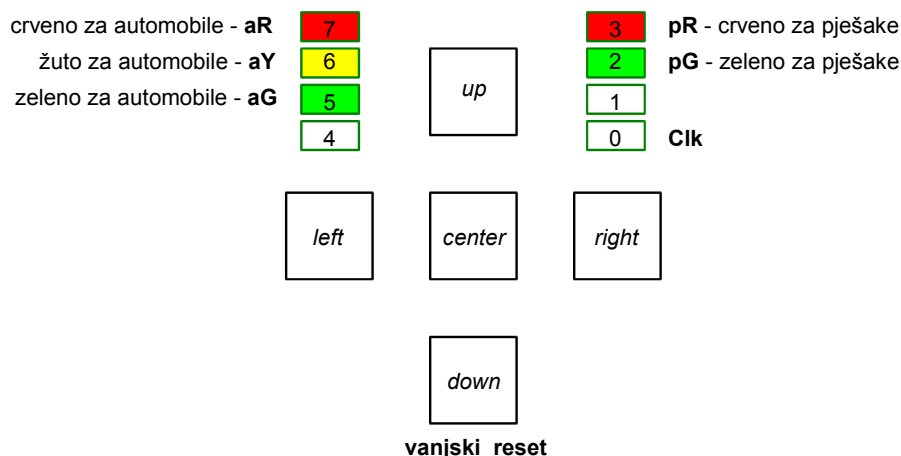
Na slici 3 prikazan je dijagram stanja modula `sem_automat` opisanog datotekom `sem_automat_pokvareni.vhd`, kojeg možete koristiti kao predložak za razvoj vlastitog automata.



Slika 3: dijagram stanja postojećeg predloška sklopa `sem_automat_pokvareni`

Povezivanje ulaznih i izlaznih signala pločice ULX2S i modula `sem_automat` izvedeno je modulom `sem_toplevel`. Modul sadrži i djelitelj takta koji iz ulaznog signala `clk_25m` frekvencije 25 MHz generira signal takta `clk` frekvencije cca. 1.5 Hz. Način

povezivanja izlaznih signala s LED indikatorima, odnosno ulaznog signala `vanjski_reset` s tipkom na FPGA pločici ULX2S ilustriran je slikom 4.



Slika 4: ulazni i izlazni signali modula `sem_toplevel` na pločici ULX2S

## 1.2 Priprema

Datoteke `sem_toplevel.vhd` i `sem_automat.vhd` dohvatite s web sjedišta laboratorijskih vježbi, uključite u novi projekt, te sintetizirajte konfiguracijsku datoteku (bitstream) za razvojnu pločicu ULX2S. **Ispitajte rad sklopa i usporedite ga s dijagramom stanja sa slike 3.** Proučite VHDL kod i u njemu identificirajte sve module označene na slici 2: registre, ROM memoriju, ostale kombinaijske module, interne signale koji ih povezuju, te vanjske signale.

U tablici 1 prikazan je željeni raspored stanja kroz koje **ispravni** automat za upravljanje semaforom treba prolaziti:

Stanje	Izlazi	Trajanje
S0	svijetli samo žuto svjetlo za automobile	1 T
S1	ne svijetli ni jedno signalno svjetlo	1 T
S2	svijetli žuto svjetlo za automobile i crveno za pješake	4 T
S3	svijetle crvena svjetla za automobile i za pješake	3 T
S4	svijetli crveno svjetlo za automobile i zeleno svjetlo za pješake	<b>X1</b> T
S5	svijetle crvena svjetla za automobile i za pješake	4 T
S6	svijetle crveno i žuto svjetlo za automobile i crveno za pješake	3 T
S7	svijetli zeleno svjetlo za automobile i crveno za pješake	<b>X2</b> T

Tablica 1: željena stanja automata i njihovo trajanje izraženo u ciklusima takta

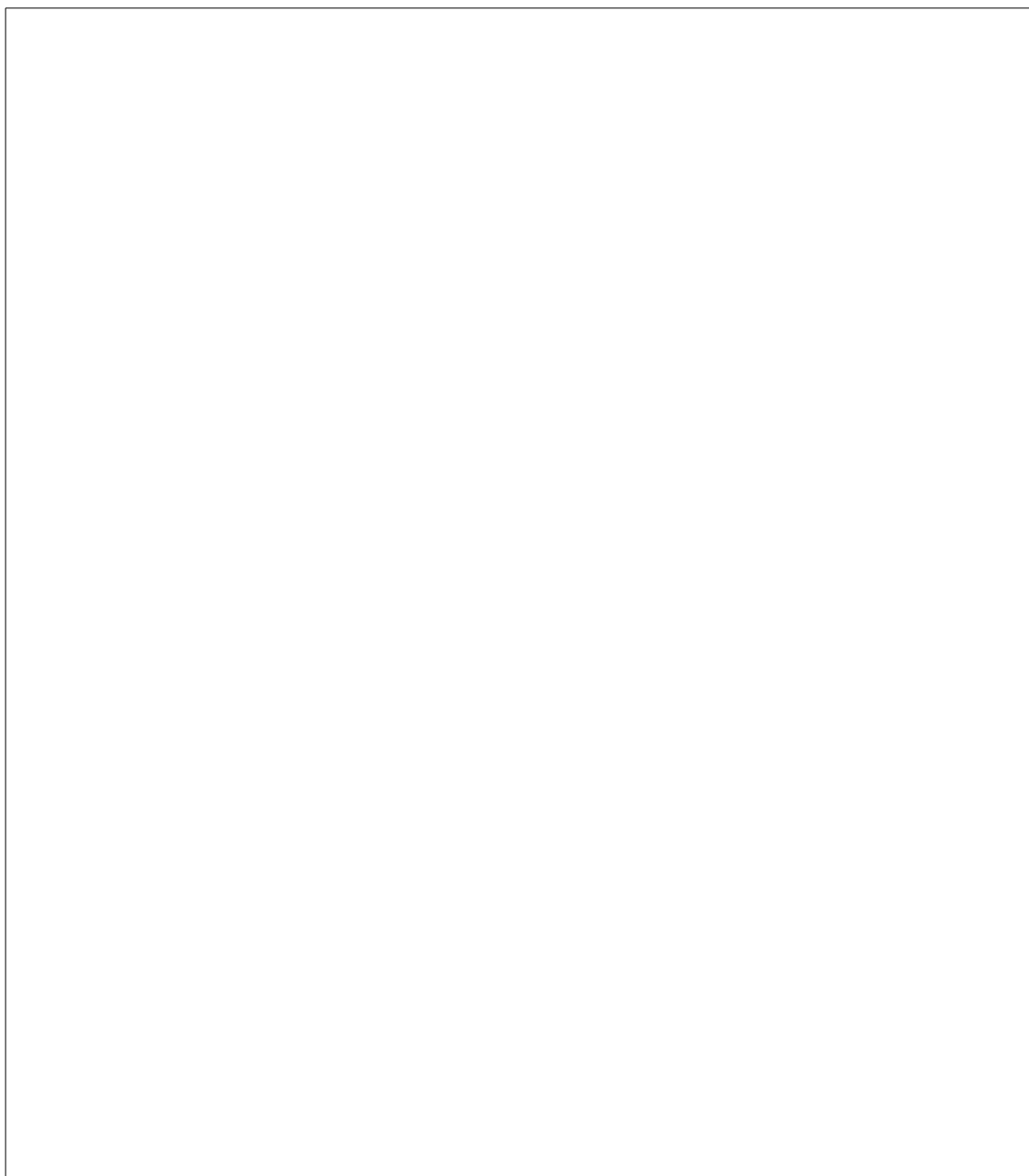
Ukoliko je ulazni signal `vanjski_reset` postavljen na 1, automat se treba prebacivati između prva dva stanja određena tablicom (S0 i S1). Kad je ulazni signal `vanjski_reset` postavljen na 0, automat treba slijedno prolaziti kroz preostalih šest stanja, zadržavajući se u svakom stanju onoliko ciklusa takta koliko je određeno brojem u drugom stupcu tablice. Vremena zadržavanja označena u tablici s **X1** i **X2** određuju se prema formulama:

$$\mathbf{X1 = (JMBAG \bmod 5) + 5}$$

$$\mathbf{X2 = (JMBAG \bmod 5) + 10}$$

Automat treba nakon zadnjeg stanja određenog tablicom (S7) prijeći u stanje S2. Ukoliko se ulazni signal `vanjski_reset` postavi na 1, automat treba prijeći u stanje S0, osim ako se već nalazi u stanju S0. Kod svakog prijelaza u novo stanje automat treba postaviti izlaz `mijenjaj_stanje` na 1, čime će *resetirati* brojač ciklusa takta.

U prazni okvir na ovoj stranici nacrtajte dijagram stanja automata u skladu s tablicom 1, te označite sve prijelaze, ulaze i izlaze automata.



## 1.3 Projektiranje automata i ispitivanje rada

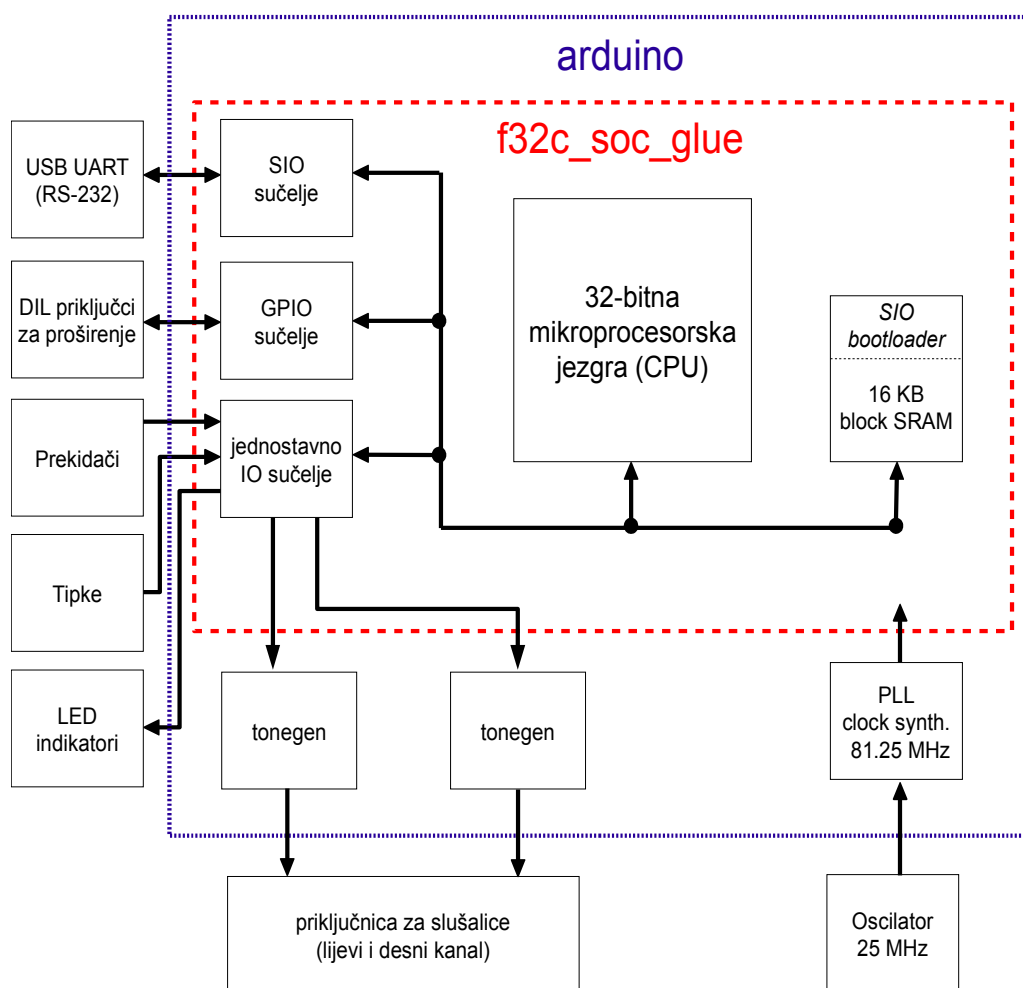
Prema dijagramu stanja kojeg ste nacrtali na prethodnoj stranici VHDLom opišite odgovarajući automat. Ulazni signali modula trebaju biti nazvani točno `clk` i `vanjski_reset`, a izlazi `aR`, `aY`, `aG`, `pR` i `pG`. Svi signali u sučelju modula trebaju biti tipa `std_logic`. Pri izradi modula možete koristiti predložak `sem_automat_pokvareni.vhd`, ali automat možete modelirati i prema vlastitim zamislima, uz uvjet da funkcijski odgovara zadanim specifikacijama, te da je izveden u samo jednom VHDL modulu odnosno datoteci.

Stvorite novi projekt u razvojnom alatu Lattice Diamond, u kojeg uključite već gotovu datoteku `sem_toplevel.vhd`, vlastitu datoteku `sem_automat.vhd`, te kao i uvijek datoteku `ulx2s.lpf`.

Prilikom ispitivanja sklopa prekidačem `sw(0)` može se uključiti odnosno isključiti prikaz otkucaja signala takta na indikatoru `led(0)`, što može olakšati praćenje vremenskih intervala zadržavanja u pojedinim stanjima automata.

Nakon što ste sintetizirali i ispitati rad sklopa na FPGA pločici, u sustav Ferko prebacite (uploadajte) datoteke `sem_automat.vhd` i `lab6.jed`.

## 2 FPGA kao mikroprocesorski sustav (1 bod)



Slika 5: Blok-shema mikroprocesorske konfiguracija FPGA sklopa

Kao podloga za vježbu pripremljen je projekt s opisom 32-bitnog mikroprocesorskog sustava. Projekt sa svim pripadajućim modulima spreman za sintezu korištenjem razvojne okoline Lattice Diamond dostupan je u obliku ZIP arhive na web sjedištu laboratorijskih vježbi. Glavni (top-level) modul projekta definiran je u datoteci `arduino.vhd`, a sastoji se od mikroprocesorskog sustava (`f32c_soc_glue`) povezanog s vanjskim priključnicama FPGA sklopa, generatora takta frekvencije 81.25 Mhz, te dvije instance modula `tonegen` koji zavisno od kodnih riječi na svojim ulazima generiraju dva tonfrekvencijska signala konstantnih frekvencija sprovedena kao izlaze na lijevi odnosno desni kanal stereo priključnice za slušalice razvojne pločice.

Za uređivanje i prevođenje programa za mikroprocesorski sustav koristit će se razvojna okolina Arduino ([www.arduino.cc](http://www.arduino.cc)). Arduino omogućuje razvoj jednostavnih programa za razne mikroprocesorske arhitekture, od "malih" 8- i 16-bitnih mikrokontrolera do složenih 32-bitnih mikroprocesorskih sustava. Programira se korištenjem podskupa jezika C++, pri čemu se u praksi može koristiti i isključivo jezik C, kojim ste u opsegu dovoljnom za izvođenje ove vježbe već ovladali u okviru kolegija Programiranje i programsko inženjerstvo. Grafičko sučelje razvojne okoline omogućuje uređivanje programa te odabir ciljane mikroprocesorske arhitekture odnosno sklopovskog sustava za koju će se izvorni kod prevoditi u strojni. Prevođenje programa obavlja se pomoću standardnih GNU alata (`gcc` / `g++`, `binutils`) koji su integrirani u razvojnu okolinu, a čiji se odabir, podešavanje i pokretanje obavlja kroz grafičko sučelje.

## 2.1 Instaliranje razvojne okoline Arduino

---

S web sjedišta [www.arduino.cc](http://www.arduino.cc) dohvatite instalacijski paket za operacijski sustav Vašeg računala te ga instalirajte na računalo. Ovaj korak možete preskočiti ako već od ranije imate instaliran Arduino verzije 1.6.4 ili novije, kao što je slučaj sa stolnim računalima u fakultetskom laboratoriju. Pokrenite Arduino, otvorite izbornik *File* → *Preferences*. U polju "Additional Boards Manager URLs:" upišite [http://www.nxlab.fer.hr/fpgarduino/package\\_f32c\\_core\\_index.json](http://www.nxlab.fer.hr/fpgarduino/package_f32c_core_index.json).

Otvorite izbornik *Tools* → *Board*: → *Boards Manager*, te pri dnu popisa raznih platformi pronađite *FPGAduino (FER + RIZ + Radiona)*, kliknite na *FPGAduino*, pokrenite *Install*. Postupak dohvata i instaliranja *FPGAduino* proširenja može potrajati nekoliko minuta.

Pod *Tools* → *Board* odaberite **FER ULX2S u varijanti s 16K BRAM**.

Pod *Tools* → *Programmer* odaberite *ujprog (FPGAduino)*.

Pod *Tools* → *Port* odaberite serijsko sučelje (*COM port*) kojeg je računalo dodijelilo priključenoj FPGA razvojnoj pločici.

## 2.2 Ispitivanje rada mikroprocesorskog sustava

---

Korištenjem razvojne okoline Lattice Diamond sintetizirajte konfiguraciju FPGA sklopa koristeći nepromijenjen predložak projekta dohvaćen u obliku ZIP arhive s web sjedišta laboratorijskih vježbi. Prije pokretanja postupka sinteze provjerite i po potrebi podesite tip ciljanog FPGA sklopa u skladu s tipom ugrađenim na Vašu razvojnu pločicu (Lattice XP2-5E ili XP2-8E), klikom desne tipke miša na tip FPGA sklopa u kartici *File List* razvojne okoline Lattice Diamond.

Konfigurirajte FPGA sklop sintetiziranom mikroprocesorskom konfiguracijom korištenjem naredbe `ujprog`. Mikroprocesorski sustav najavljuje spremnost za prihvati i izvođenje programa izmjeničnim postupnim paljenjem i gašenjem LED indikatora s

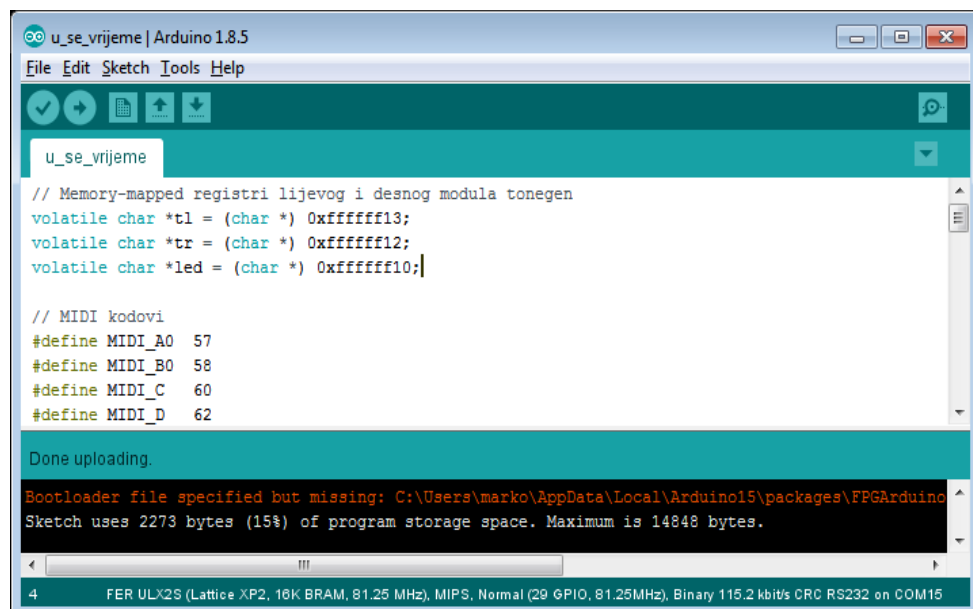
lijeve i desne strane razvojne pločice. Nakon konfiguriranja FPGA sklopa program `ujprog` treba zatvoriti (pri pozivanju **NE** koristiti opciju `-t`).

U radnoj okolini Arduino odaberite *File* → *Examples* → *01. Basics* → *Blink*. Provjerite da li ste odabrali ispravne postavke ciljane mikroprocesorske platforme, kao što je prikazano pri dnu slike 6 (FER ULX2S, 16K BRAM, MIPS), te da li ste odabrali ispravno serijsko sučelje za programiranje pločice (u primjeru sa slike 6 odabran je port *COM15*).

Prevedite program i učitajte ga na pločicu pritiskom na tipku *Upload* (ispod izbornika *Edit*). Nakon uspješnog pokretanja programa jedan LED indikator trebao bi mijenjati stanje (paliti se i gasiti) svake sekunde, kao što je zadano programom "*Blink*".

Na primjeru "*Blink*" uočite kako za razliku od uobičajene konvencije pokretanja C i C++ programa u kojima izvođenje otpočinje pozivom funkcije `main()`, Arduino programi imaju dvije glavne funkcije: `setup()`, koja ima inicijalizacijsku namjenu te se poziva samo jednom, nakon čega će se u beskraju petlji pozivati funkcija `loop()`.

Pokušajte napraviti jednostavne promjene u programu, npr. mijenjanjem kašnjenja između promjene stanja LED dioda. Prevedite promijenjeni program i učitajte ga na pločicu.



Slika 6: Arduino razvojno sučelje, primjer upravljanja modulima *tonegen*

Vanjskim ulazno-izlaznim sučeljima mikroprocesorskog sustava sa sheme 5 programski se može pristupiti čitanjem odnosno pisanjem na adrese koje su unaprijed zadane izvedbom mikroprocesorske konfiguracije (znatiželjni studenti definicije mogu pronaći u modulu `f32_soc_glue.vhd`). Konkretno, na LED indikatorima bit će vidljiv oktet (byte) upisan na adresu `0xffffffff10`, dok će okteti upisani na adrese `0xffffffff12` i `0xffffffff13` biti sprovedeni na ulaze modula *tonegen*, čiji su pak izlazi povezani s lijevim odnosno desnim kanalom priključnice za slušalice.

Kako bi iz C programa mogli pisati u ove adresne raspone, potrebno je deklarirati varijable pokazivače željenog tipa, te iste ispravno inicijalizirati. Prilikom deklariranja pokazivača na adresni raspon vanjskih registara nužno je koristiti ključnu riječ `volatile` koja će za ciljani pokazivač isključiti algoritme optimiranja kako ne bi došlo do uklanjanja pristupa memorijskom rasponu, kojeg bi C prevoditelj mogao ocijeniti suvišnim bez primjene ključne riječi `volatile`. Primjer deklaracije pokazivača vidljiv je



na slici 6, a preuzet je iz gotovog primjer C programa za Arduino `u_se_vrijeme.ino` kojeg možete naći na web sjedištu laboratorijskih vježbi. Primjer pokazuje kako upravljati LED indikatorima i generatorima tonfrekvencijskog signala, koje možete ispitati priključenjem slušalica u odgovarajuću priključnicu FPGA razvojne pločice.

## 2.3 Zadatak: programski upravljan semafor

---

Funkcionalnost semafora definiranu shemom sa slike 4 te tablicom 1 ostvarite programski, korištenjem platforme Arduino i mikroprocesorske konfiguracije FPGA sklopa koju ste ispitali u prethodnom koraku.

Programski ostvaren automat treba voditi "taktom" frekvencije 1 Hz. Potrebna kašnjenja u izvođenju programa ostvarite korištenjem funkcije `delay()`, čiji je jedini argument trajanje kašnjenja izraženo u milisekundama kako cijeli broj.

Ulaz za resetiranje automata semafora ostvarite čitanjem s adrese `0xffffffff00`, povezane s modulom koji omogućuje očitavanje stanja tipki razvojne pločice. Tri bita najveće težine okteta pročitane s navedene adrese uvijek će biti nula, dok će preostali bitovi odgovarati stanju tipki `btn_center` (bit 4), `btn_up` (bit 3), `btn_down` (bit 2), `btn_left` (bit 1) i `btn_right` (bit 0).

Funkcionalnost sklopa proširite zvučnom signalizacijom uobičajenom za pomoć slabovidnim osobama, na način da se uz crveno svjetlo za pješake svake sekunde oglašava kratki zvučni signal, dok se uz zeleno svjetlo za pješake kratki zvučni signal treba oglašava češće, četiri do pet puta u sekundi.

U sustav Ferko pohranite izvorni kod svog Arduino programa (`semafor.ino`).

## 2.4 Uobičajeni problemi u radu s mikroprocesorskom konfiguracijom

---

- na nekim operacijskim sustavima nakon instaliranja razvojnog okruženja Arduino za njegov daljnji ispravan rad potrebno je prethodno restartati računalo;
- odabire se pogrešna konfiguracija mikroprocesorskog sustava, npr. 1M SRAM umjesto 16K BRAM, RISC-V umjesto MIPS;
- prije odašiljanja (upload) Arduino programa nužno je konfigurirati FPGA sklop mikroprocesorskom konfiguracijom korištenjem alata uprog; ako na pločici naizmjenice ne tinjaju lijevi i desni LED indikator ne će biti moguće izvršiti upload, pri čemu će Arduino prijaviti grešku "TX error at 00000400";
- greška "TX error at 00000400" javlja se i ako je odabrano pogrešno serijsko sučelje (COM port) prema razvojnoj pločici: kroz izbornik Tools -> Port odaberite ispravno serijsko sučelje