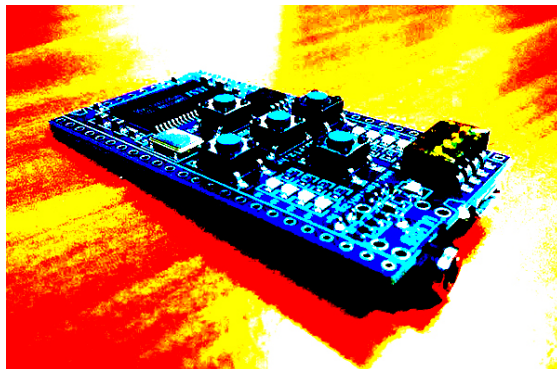


Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva



Upute za rad s FPGA razvojnou pločicom ULX2S

Marko Zec

listopad 2017.

Sadržaj

1	Uvod.....	2
1.1	Programirljiva polja logičkih blokova (FPGA).....	3
1.2	Postupci opisa, sinteze i programiranja konfiguracije FPGA sklopa.....	5
2	Komponente razvojne pločice.....	6
2.1	Napajanje.....	6
2.2	FPGA sklop.....	7
2.3	Sučelje JTAG.....	7
2.4	Tipke.....	8
2.5	Prekidači.....	8
2.6	LED indikatori.....	9
2.7	Višenamjenska stereo priključnica.....	9
2.8	Sučelje RS-232.....	10
2.9	Generator takta.....	10
2.10	Statička memorija (RAM).....	11
2.11	Flash memorija.....	11
2.12	MicroSD kartica.....	12
2.13	DIL priključnice za proširenje.....	12
2.14	Definicije vanjskih signala.....	13
3	Instaliranje programske podrške.....	13
3.1	Instaliranje programskog paketa Lattice Diamond.....	13
3.2	Instaliranje USB pogonskih programa (<i>drivera</i>) za razvojnu pločicu.....	13
3.3	Instaliranje programskog alata za programiranje pločice.....	14
4	Rad u razvojnom okruženju Lattice Diamond.....	15
4.1	Stvaranje novog projekta.....	15
4.2	Shematski opis digitalnog sklopa.....	16
4.3	Sinteza konfiguracije FPGA sklopa.....	20
4.4	Programiranje FPGA sklopa.....	20
4.5	Uobičajeni problemi u radu s FPGA razvojnim alatima.....	22
5	Opis sklopova jezikom VHDL.....	24
5.1	Multipleksor.....	27
5.2	Dekoder.....	28
6	Razvojna pločica kao samostalni mikroprocesorski sustav.....	28
6.1	Programiranje u razvojnom okruženju Arduino.....	28
6.2	Programiranje pomoću GNU razvojnih alata (C / C++).....	29

1 Uvod

Pločica ULX2S (USB, Lattice XP2, SRAM) namijenjena je razvoju i ispitivanju rada digitalnih sklopova i sustava koji se mogu sintetizirati na integriranim programirljivim poljima logičkih blokova (eng. *Field Programmable Gate Array – FPGA*). Prvenstveno je zamišljena kao nastavno pomagalo u svladavanju temeljnih načela rada, analize i projektiranja digitalnih sklopova, ali zbog svojih malih dimenzija i mogućnosti povezivanja s vanjskim sklopovima može poslužiti i kao modul širokog spektra primjene za ugradnju u kompleksnije uređaje. Sintezom odgovarajuće procesorske jezgre na FPGA sklopu razvojna pločica može raditi i kao samostalni programski upravljiv mikroprocesorski sustav.

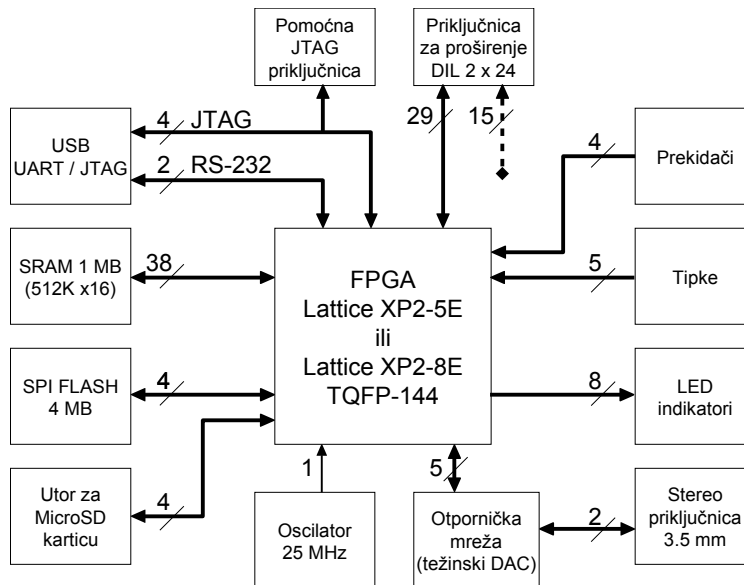
Glavna komponenta na pločici je FPGA sklop XP2 proizvođača Lattice Semiconductor s oko 5000 (varijanta XP2-5E) ili 8000 (varijanta XP2-8E) programirljivih logičkih elemenata temeljenih na preglednim tablicama s 4 ulaza (eng. *4-input lookup table – LUT*). FPGA sklop se može neograničeni broj puta rekonfigurirati putem USB sučelja, koje ujedno služi i kao glavni izvor napajanja, te kroz koje se može ostvariti i asinkrona serijska komunikacija s računalom (RS-232). Kako osim USB kabela ne zahtijeva nikakve dodatne komponente za napajanje i povezivanje s računalom, s pločicom se može raditi kako u laboratoriju tako i kod kuće. Robusni mehanički dizajn i male dimenzije čine pločicu podesnom za učestalo prenošenje.

Ugrađeni FPGA sklop raspolaže s ukupno 100 programirljivih vanjskih ulazno-izlaznih priključaka putem kojih je ostvareno povezivanje sa sljedećim komponentama i priključnicama na razvojnoj pločici:

- LED indikatori (8)
- Tipke (5)
- Prekidači u *dual in-line (DIL)* kućištu (4)
- Generator takta (oscilator) frekvencije 25 Mhz i stabilnosti +/- 25 ppm
- USB - RS-232 asinkrono serijsko sučelje
- Višenamjenska stereo priključnica s 4-bitnom težinskom otpornom mrežom za D/A pretvorbu
- Utor za memorijske kartice MicroSD
- Flash memorija sa serijskim sučeljem (SPI) kapaciteta 4 MByte
- Statička RAM memorija kapaciteta 1 MByte (512k * 16 bit)
- Priključnice za proširenje: ukupno 2 * 24 priključka u standardnom rasteru 2.54 mm, od kojih su 44 signala povezana na FPGA sklop

Na razvojnoj pločici predviđeno je i mjesto za ugradnju pomoćnog priključka za programiranje (JTAG), koji može poslužiti za povezivanje vanjskog programatora u slučaju potrebe za funkcijama koje trenutno nisu podržane na integriranom USB-JTAG sučelju (npr. *in-system debugging*).

Način povezivanja FPGA sklopa s komponentama i priključnicama na razvojnoj pločici prikazan je blok shemom na slici 1.

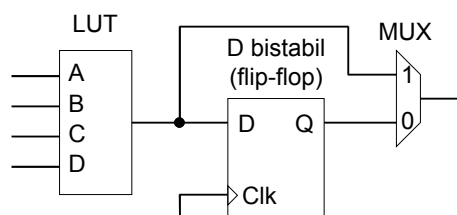


Slika 1: blok-shema povezivanja komponenti razvojne pločice

1.1 Programirljiva polja logičkih blokova (FPGA)

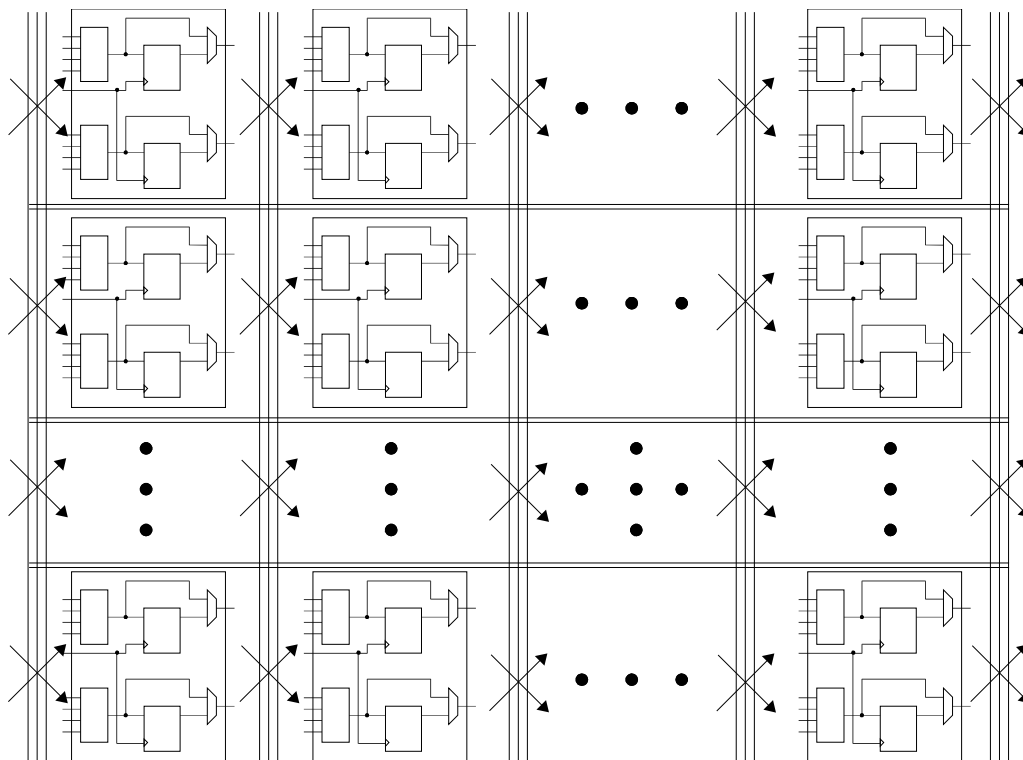
Programirljiva polja logičkih blokova (*eng. Field programmable gate array – FPGA*) su digitalni sklopovi vrlo visokog stupnja integracije u kojima se veliki broj relativno jednostavnih i uniformnih programirljivih logičkih elemenata može povezati u proizvoljni kompleksni sklop, te takav sklop povezati s vanjskim svijetom putem programirljivih ulazno-izlaznih priključaka. FPGA sklopovi danas imaju široku primjenu u industriji (automatsko upravljanje), medicini (ultrazvuk, CT, MR), prometu i obrambenim sustavima (sonar, radar), radiokomunikacijama i telekomunikacijama (odašiljači i bazne stanice, mrežni prospojnici, sklopovski vatrozidovi), potrošačkoj elektronici (npr. HD video prospojnici), te općenito pri razvoju digitalnih sustava kao pomagala za ispitivanje prototipnih implementacija.

Temeljne komponente programirljivih blokova su **pregledna tablica** (*eng. lookup table – LUT*) i sinkroni bridom okidani **D bistabil** (*eng. flip-flop*). Pregledna tablica omogućuje izvedbu proizvoljne logičke funkcije od N varijabli, gdje je N broj ulaza, koji u današnje vrijeme zavisno od proizvođača i kategorije FPGA sklopa može biti 4, 5 ili 6. Izlaz iz jedne pregledne tablice može se direktno dovesti na ulaz memorijskog elementa (bistabil) ili povezati s ulazima drugih preglednih tablica te takvim kaskadiranjem ostvariti kompleksnije logičke funkcije. Pojednostavljeni model temeljnog programirljivog **logičkog elementa** (*eng. logic element*) modernog FPGA sklopa koji se sastoji od pregledne tablice i D bistabila prikazan je na slici 2.



Slika 2: struktura programirljivog logičkog elementa

Više logičkih elemenata (obično 2 ili 4) grupirani su u tzv. **logičke blokove**, za koje svaki proizvođač ima vlastitu (ponekad nekonzistentnu) terminologiju, npr. *configurable logic block*, *logic slice*, *logic cell* i slično. Za međusobno povezivanje logičkih blokova FPGA sklopovi raspolažu s programirljivom mrežom prospoynih vodova koja omogućuje povezivanje bilo kojeg izlaza bilo kojeg logičkog bloka s bilo kojim ulazom nekog drugog ili istog logičkog bloka odnosno elementa. Posebni dio mreže služi za sinkrono dovođenje signala takta do svakog logičkog bloka. Pojednostavljena struktura tipičnog FPGA sklopa koja se sastoji od polja programirljivih logičkih blokova i prospoynje mreže prikazana je na slici 3.



Slika 3: struktura programirljivog polja logičkih blokova

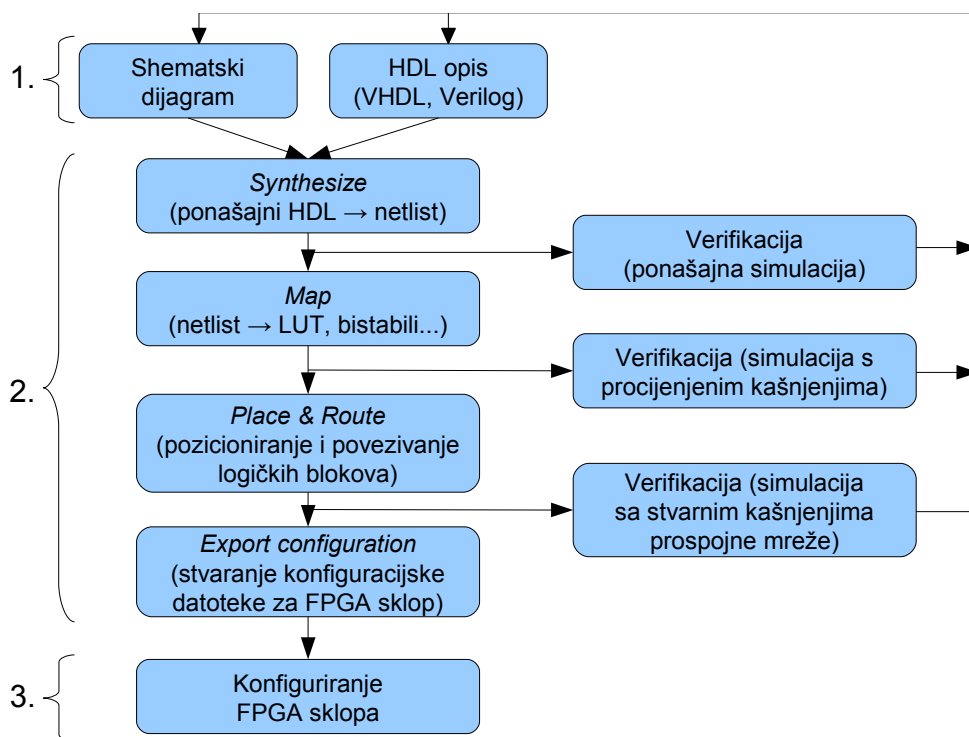
Veličine polja današnjih FPGA sklopova kreću su u rasponu od oko 1.000 do 2.000.000 preglednih tablica (LUT) odnosno logičkih elemenata. Kašnjenja pojedine pregledne tablice (LUT) tipično su reda veličine od 100 do 300 ps, kašnjenja bistabila su reda veličine od 200 do 500 ps, dok kašnjenja koja unosi prospoyna mreža mogu doseći i do nekoliko ns. Bitno je uočiti da svaki logički element FPGA sklopa radi paralelno i potpuno nezavisno od drugih, za razliku od računala opće namjene u kojima jezgra mikroprocesora slijedno izvršava programske instrukcije jednu za drugom.

Uz velik broj generičkih logičkih blokova temeljenih na preglednim tablicama (LUT) te posebnih elemenata vezanih uz programirljiva vanjska ulazno-izlazna sučelja, moderni FPGA sklopovi obično raspolažu i s manjim brojem dodatnih specijaliziranih elemenata kao što su blokovi statičke RAM memorije, blokovi za množenje, sintetizatori takta, blokovi za izvedbu sučelja prema vanjskim dinamičkim RAM memorijama (DDR), brza serijska sučelja (SERDES), interna Flash memorija i slično. Sve današnje FPGA platforme raspolažu i sa specijaliziranim unutarnjim putevima koji povezuju susjedne logičke blokove, a koji su prvenstveno namijenjeni brzom prijenosu bita preljeva kod implementacije višebitnih blokova za zbrajanje ili oduzimanje (tzv. *fast carry chain*).

1.2 Postupci opisa, sinteze i programiranja konfiguracije FPGA sklopa

Konfiguracija FPGA sklopova, dakle logičkih blokova i prospojne mreže, u pravilu se generira strojno uz pomoću specijaliziranih programskih alata, tzv. sintetizatora (*eng. synthesis tools*), a na temelju opisa sklopa specificiranog shemom ili korištenjem jezika za opis digitalnih sklopova (*eng. hardware description language - HDL*). Kako svaka kategorija FPGA sklopova svakog proizvođača ima različitu internu strukturu i arhitekturu, tako se razlikuju i alati za sintezu, a vlastite algoritme za sintezu konfiguracija svojih FPGA sklopova svaki proizvođač brižno čuva od konkurencije. Drugim riječima, za projektiranje digitalnih sklopova i sustava koji će se sintetizirati na FPGA sklopovima **nije potrebno poznavati implementacijske detalje ciljane FPGA platforme**, ali je korisno razumjeti njihovu strukturu kako bi se opis željenog sklopa formulirao na način koji omogućuje sintetizatoru stvaranje optimalne konfiguracije logičkih blokova i prospojne mreže.

Rad s programskim alatima za sintezu konfiguracije FPGA sklopova može se podijeliti u više cjelina, kao što je prikazano slikom.



Slika 4: koraci pri opisu sklopa, sinteze konfiguracije i programiranja FPGA sklopa

Prvi korak je opis željenog sklopa na način koji je prikladan za daljnju strojnu obradu, najčešće unosom shematskog dijagrama ili korištenjem jezika za opis digitalnih sklopova, od kojih su danas najšire prihvaćeni i podržani VHDL i Verilog.

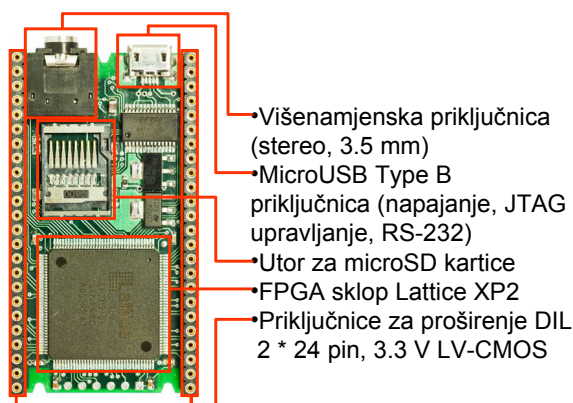
Sinteza konfiguracije FPGA sklopa na temelju shematskog ili HDL opisa provodi se u više koraka. U koraku koji je u većini programskih alata nazvan "Synthesize" analizira se opis sklopa te se ponašajno opisani blokovi zamjenjuju funkcijski ekvivalentnim strukturnim modelima izgrađenim od primitiva specifičnih za ciljanu FPGA platformu, pri čemu se provodi i automatizirana optimizacija kombinacijske i sekvencijske logike. Rezultirajuća mreža primitiva, koja se uobičajeno naziva *netlist*, može se analizirati i funkcijski verificirati korištenjem odgovarajućeg simulatora, pri čemu se ne modeliraju kašnjenja primjenjenih primitiva.

U sljedećem koraku, koji se uobičajeno naziva "*Map*", sintetizator primitive (npr. AND_2, OR_8, MUX_16_1) zamjenjuje mrežom preglednih tablica (LUT), memorijskih elemenata te ostalih specijaliziranih logičkih blokova koji odgovaraju ciljanoj FPGA platformi. U ovom koraku sintetizator može napraviti i simulacijski model sklopa koji uključuje grubu procjenu kašnjenja pri propagaciji signala, budući da u ovom koraku još nisu poznata kašnjenja prospojne mreže.

U koraku nazvanom "*Place and Route*" sintetizator raspoređuje logičke blokove na FPGA sklopu i povezuje ih na način koji će omogućiti ispravan rad sklopa na najvišoj mogućoj frekvenciji takta. Zavisno od kompleksnosti opisanog sklopa, ciljane frekvencije takta, postavki algoritma *place and route* algoritma te brzine računala na kojem se izvodi algoritam, ovaj dio postupka sinteze može potrajati od desetak sekundi do više sati. Po završetku ovog koraka sintetizator može napraviti simulacijski model koji uključuje vrlo preciznu procjenu kašnjenja svih logičkih elemenata i prospojne mreže. Analizom takvog modela mogu se pronaći kritični putevi projektiranog sklopa koje ograničuju brzinu rada, na temelju čega se može pristupiti poboljšanju opisa sklopa.

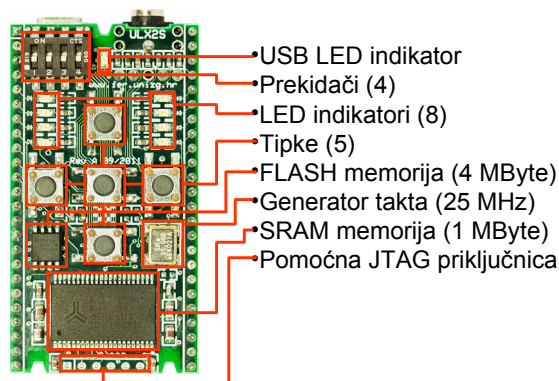
Krajnji rezultat procesa sinteze je konfiguracijska datoteka (*eng. configuration bitstream*) kojom se pomoću odgovarajućeg alata može programirati FPGA sklop, odnosno pohraniti u njegovu trajnu memoriju (Flash) iz koje će sklop automatski učitati konfiguraciju prilikom uspostave napajanja.

2 Komponente razvojne pločice



- Višenamjenska priključnica (stereo, 3.5 mm)
- MicroUSB Type B priključnica (napajanje, JTAG upravljanje, RS-232)
- Utor za microSD kartice
- FPGA sklop Lattice XP2
- Priključnice za proširenje DIL 2 * 24 pin, 3.3 V LV-CMOS

Slika 5: donja strana pločice



- USB LED indikator
- Prekidači (4)
- LED indikatori (8)
- Tipke (5)
- FLASH memorija (4 MByte)
- Generator takta (25 MHz)
- SRAM memorija (1 MByte)
- Pomoćna JTAG priključnica

Slika 6: gornja strana pločice

Na slikama 5 i 6 prikazan je fizički raspored komponenti na pločici ULX2S. Dimenzije pločice su približno 64 * 36 * 9 mm. Pločica se s računalom povezuje putem priključnog kabela tipa USB A na micro-USB tip B, koji se ne isporučuje zajedno s pločicom.

2.1 Napajanje

Pločica je projektirana za napajanje putem USB priključka ili uz određena ograničenja putem DIL podnožja za proširenje. Nominalni napon napajanja na USB priključku je 5 V, a pločica će ispravno raditi uz ulazni napon u rasponu od 4.3 V do 6.3 V. Dovođenje napona višeg od 6.3 V na USB priključku može rezultirati oštećenjem komponenti.

Na pločicu su ugrađeni linearni regulatori napajanja s izlazima 3.3 V i 1.2 V. Napon od 3.3 V koristi se za napajanje svih ulazno-izlaznih sučelja FPGA sklopa, generatora

takta, SRAM i Flash memorije, te microSD kartice, dok jezgra FPGA sklopa radi na 1.2 V. Ukupna potrošnja struje zavisi od kompleksnosti konfiguracije i frekvenciji takta unutar FPGA sklopa, te o statičkom i dinamičkom opterećenju izlaznih signala, a uobičajeno se može kretati u rasponu od 55 mA do 300 mA ili više. Kako linearni regulatori razliku između ulaznog i izlaznog napona pretvaraju u toplinsku energiju koja je proporcionalna struji kroz regulator, kod kompleksnijih konfiguracija FPGA sklopa regulatori napajanja pa i FPGA sklop mogu se osjetno zagrijati, što međutim ne utječe na stabilnost njihovog rada i na vijek trajanja. Na primjer, uz ispitnu konfiguraciju FPGA sklopa razvojna pločica tipično troši između 170 i 210 mA, što pri naponu napajanja od 5V odgovara ukupnoj disipaciji snage od otprilike 1 W.

Pločica nema ugrađenu vlastitu zaštitu od eventualnog kratkog spoja, već se oslanja na zaštitne mehanizme ugrađene u USB sučelja na računalima koja ograničavaju potrošnju priključenih uređaja na najviše 500 mA. Ako potrošnja USB uređaja premaši razinu od 500 mA računalo će automatski privremeno prekinuti napajanje bez oštećenja USB sučelja ili priključenog uređaja. Međutim, dodatni oprez potreban je kod priključenja pločice na računalo putem USB *huba* s vlastitim napajanjem ili na nezavisne izvore napajanja s USB priključkom koji se uobičajeno koriste za punjenje baterija mobilnih telefona i sličnih uređaja potrošačke elektronike, budući da takvi uređaji često nemaju ugrađene sklopove za zaštitu od kratkog spoja.

2.2 FPGA sklop

Na pločicu ULX2S ugrađuje se FPGA sklop Lattice XP2 u varijanti 5E ili 8E. Razlike između ove dvije varijante FPGA sklopa prikazane su tablicom:

	XP2-5E	XP2-8E
Preglednih tablica (LUT)	oko 5000	oko 8000
Raspodijeljeni RAM (Kbits)	10	18
Zasebnih RAM blokova (2048x9 bits)	9	12
Blokova za množenje (18x18)	12	16
Blokova za sintezu signala takta (PLL)	2	2

Glavna značajka po kojoj se sklop Lattice XP2 razlikuje od većine FPGA sklopova drugih proizvođača je ugrađena Flash memorija za konfiguraciju. Po uspostavi napajanja FPGA sklop će automatski konfiguracijskim zapisom iz ugrađene Flash memorije programirati vlastiti konfiguracijski RAM, što omogućuje spremnost sklopa za rad samo nekoliko desetaka mikrosekundi nakon uključanja napajanja. Ugrađenu konfiguracijsku Flash memoriju moguće je programirati do 10.000 puta, a deklarirano najkraće vrijeme pamćenja Flash memorije je 20 godina. Za potrebe ispitivanja rada konfiguraciju FPGA sklopa moguće je programirati putem JTAG sučelja i direktno u konfiguracijski RAM bez ograničenja broja ciklusa pisanja.

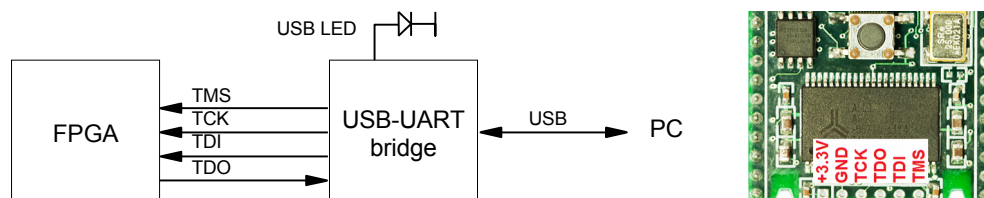
Napajanje V_{CCIO} svih ulazno-izlaznih sučelja je povezano je na izlaz linearnog regulatora 3.3 V, što znači da su na pločici ULX2S svi priključci FPGA sklopa konfigurirani su za rad s naponskim razinama između 0 i 3.3 V, o čemu treba voditi računa pri eventualnom povezivanju pločice s vanjskim uređajima.

2.3 Sučelje JTAG

FPGA sklop programira se putem sučelja JTAG koje je na pločici ULX2S izvedeno

pomoću USB-UART prilagodnika FT-232R. JTAG signalima TMS, TCK, TDI i TDO upravlja se u pravilu pomoću programskog alata `ujprog` kojeg se pokreće na računalu na koje je priključena razvojna pločica (slika 7). Za vrijeme programiranja FPGA sklopa USB LED indikator trepće frekvencijom približno 1 Hz.

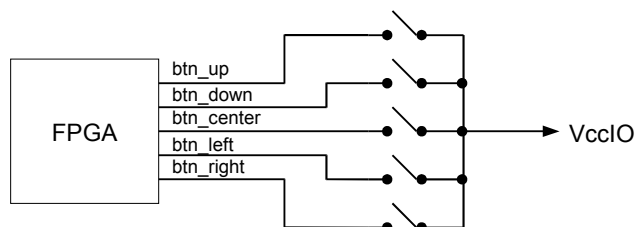
Priključci USB-UART prilagodnika povezani sa signalima TMS, TCK, TDI i TDO su postavljeni u stanje visoke impedancije osim za vrijeme izvršavanja programskog alata `ujprog`, što prema potrebi omogućuje i povezivanje s vanjskim uređajem za programiranje putem pomoćne JTAG priključnice.



Slika 7: Signali i pomoćni priključci sučelja JTAG

2.4 Tipke

Pritisak na svaku od pet tipki na razvojnoj pločici (`btn_up`, `btn_down`, `btn_center`, `btn_left`, `btn_right`) povezuje odgovarajuću priključnicu FPGA sklopa direktno na visoku naponsku razinu (V_{CCIO}). Kad tipke nisu pritisnute, signali će biti pritegnuti niskoj naponskoj razini putem *pull-down* otpornika integriranih u FPGA sklop, a primjena kojih je određena zapisom *I/O preference file*. Shema povezivanja tipki s FPGA sklopom prikazana je na slici 8:



Slika 8: tipke

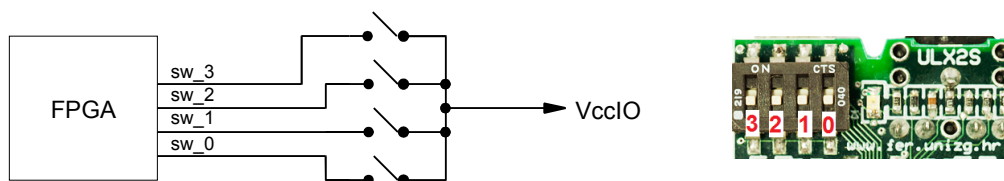
Deklaracija signala u sučelju sklopa (VHDL):

```
port (
  btn_left: in std_logic;
  btn_right: in std_logic;
  btn_up: in std_logic;
  btn_down: in std_logic;
  btn_center: in std_logic
);
```

2.5 Prekidači

U položaju "ON" svaki od četiri mikroprekidača povezuje odgovarajuću priključnicu FPGA sklopa direktno na visoku naponsku razinu (V_{CCIO}). U položaju "OFF" signali će biti pritegnuti na nisku naponsku razini putem *pull-down* otpornika integriranih u FPGA sklop, primjena kojih je određena zapisom *I/O preference file*.

Gledano s gornje strane pločice, s lijeve na desnu stranu, prekidači su numerirani slijedećim redom: sw_3, sw_2, sw_1, sw_0. Shema povezivanja mikroprekidača s FPGA sklopom prikazana je na slici 9:



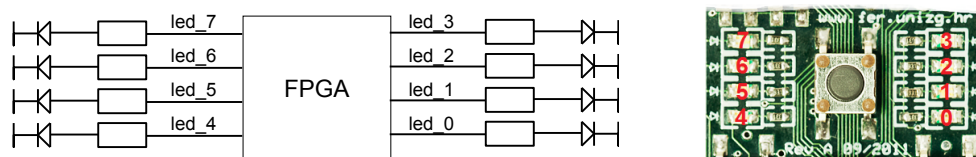
Slika 9: mikroprekidači

Deklaracija signala u sučelju sklopa (VHDL):

```
port (
  sw: in std_logic_vector(3 downto 0)
);
```

2.6 LED indikatori

LED indikatori povezani su s FPGA sklopom putem otpornika za ograničavanje struje prema shemi sa slike 10. Gledano s gornje strane pločice, LED indikatori na lijevoj strani odnosno njima pripadajući signali numerirani su slijedećim redom: led_7, led_6, led_5 i led_4, a indikatori na desnoj strani pločice su led_3, led_2, led_1 i led_0.

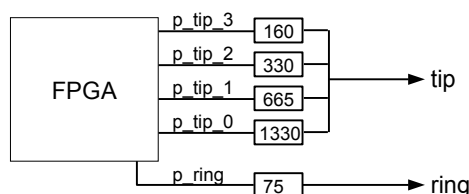


Slika 10: LED indikatori

Deklaracija signala u sučelju sklopa (VHDL):

```
port (
  led: out std_logic_vector(7 downto 0)
);
```

2.7 Višenamjenska stereo priključnica



Slika 11: težinska otporna mreža na stereo priključnici

Stereo utičnica prvenstveno je namijenjena za priključivanje niskoomskih slušalica ili ekrana s kompozitnim video ulazom, ali može poslužiti i za druge svrhe. Na kontakt na vrhu priključnice ("tip") dovodi se signal s 4-bitne težinske otporne mreže koja je proračunata tako da uz pretpostavljeni teret od 75 Ohma na izlazu može dati napon

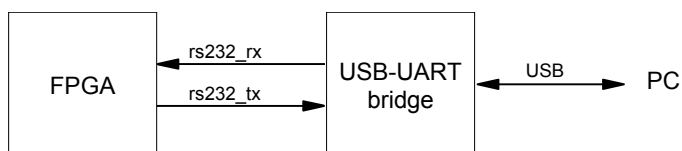
između 0 i 1.5 V u 16 diskretnih koraka. Na središnji kontakt priključnice ("ring") dovodi se samo jedan signal putem otpornika od 75 Ohma. Težinska otporna mreža koja povezuje stereo priključnicu s FPGA sklopom prikazana na slici 11.

Deklaracija signala u sučelju sklopa (VHDL):

```
port (  
  p_ring: out std_logic;  
  p_tip: out std_logic_vector(3 downto 0)  
);
```

2.8 Sučelje RS-232

Komunikacija između sklopa sintetiziranog na FPGA chipu i računala najjednostavnije se može ostvariti asinkronim serijskim prijenosom podataka putem USB-UART prenosnika ugrađenog na razvojnu pločicu. USB-UART prenosnik FT-232R podržava brzine prijenosa između 300 i 3000000 bita u sekundi, a na razvojnoj pločici se koriste samo podatkovni signali `rs232_tx` i `rs232_rx`, kao što je prikazano na slici 12:



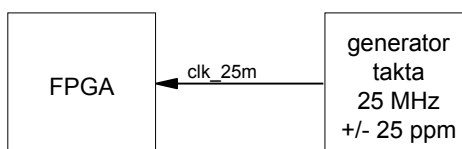
Slika 12: signali za serijsku komunikaciju s računalom

Deklaracija signala u sučelju sklopa (VHDL):

```
port (  
  rs232_tx: out std_logic;  
  rs232_rx: in std_logic  
);
```

2.9 Generator takta

Na razvojnu pločicu ugrađen je generator takta frekvencije 25 MHz i visoke stabilnosti u velikom rasponu temperaturnih, električkih i mehaničkih uvjeta. Izlazni signal generatora takta na sučelju prema FPGA sklopu nazvan je `clk_25m` (slika 13).



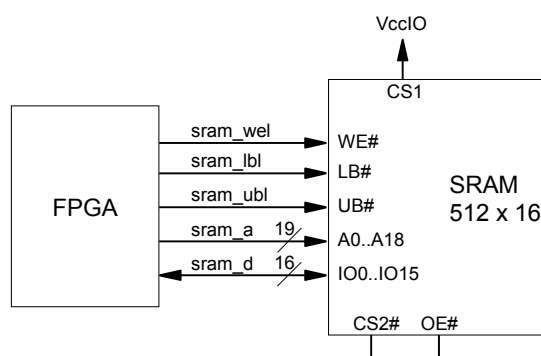
Slika 13: signal takta 25 MHz

Deklaracija signala u sučelju sklopa (VHDL):

```
port (  
  clk_25m: in std_logic  
);
```

2.10 Statička memorija (RAM)

Na razvojnu pločicu ugrađuje se statička RAM memorija (SRAM) kapaciteta 1 MByte organizirana kao polje od 524288 16-bitnih riječi. Jedan ciklus čitanja ili pisanja mora trajati najmanje 55 ili 70 ns, zavisno od varijante memorije ugrađene na pločicu. Upravljački signali CS1, CS2# i OE# na pločici ULX2S su povezani na način da su stalno aktivni, pa se radom memorije SRAM upravlja isključivo signalima WE# (*write enable low*), LB# (*lower byte enable low*) i UB# (*upper byte enable low*). Za adresiranje podataka služi 19-bitna sabirnica sram_a, a podaci se između FPGA sklopa i memorije prenose 16-bitnom sabirnicom sram_d, pri čemu smjer toka podataka određuje upravljački signal sram_wel. Shema sučelja između FPGA sklopa i memorije SRAM prikazana je slikom 14:



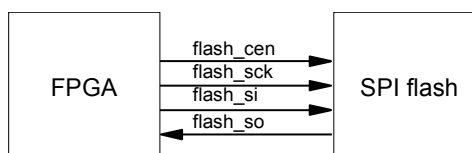
Slika 14: signali sučelja prema vanjskoj memoriji (RAM)

Deklaracija signala u sučelju sklopa (VHDL):

```
port (  
  sram_a: out std_logic_vector(18 downto 0);  
  sram_d: inout std_logic_vector(15 downto 0);  
  sram_wel: out std_logic;  
  sram_lbl: out std_logic;  
  sram_ubl: out std_logic  
);
```

2.11 Flash memorija

Komunikacija s Flash memorijom ugrađenoj na razvojnu pločicu ostvaruje se prema protokolu SPI korištenjem signala flash_cen, flash_sck, flash_si i flash_so (slika 15):



Slika 15: signali sučelja prema vanjskoj Flash memoriji

Deklaracija signala u sučelju sklopa (VHDL):

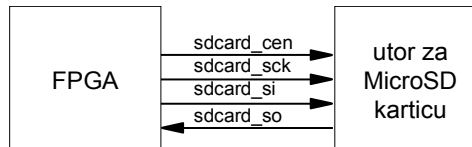
```

port (
    flash_so: in std_logic;
    flash_cen: out std_logic;
    flash_sck: out std_logic;
    flash_si: out std_logic
);

```

2.12 MicroSD kartica

Komunikacija s MicroSD memorijskom karticom ostvaruje se prema protokolu SPI korištenjem signala `sdcard_cen`, `sdcard_sck`, `sdcard_so` i `sdcard_si` (slika 16):



Slika 16: signali sučelja prema MicroSD kartici

Deklaracija signala u sučelju sklopa (VHDL):

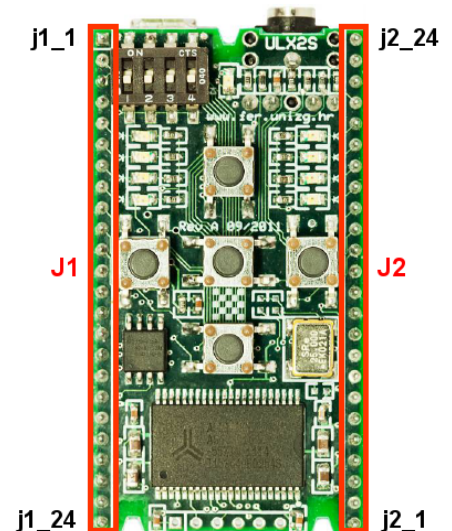
```

port (
    sdcard_so: in std_logic;
    sdcard_cen: out std_logic;
    sdcard_sck: out std_logic;
    sdcard_si: out std_logic
);

```

2.13 DIL priključnice za proširenje

S lijeve i desne strane razvojne pločice nalazi se po jedan stupac slobodnih kontakata u standardnom rasteru 2.54 mm na koje se mogu zalemiti spojnice za povezivanje s vanjskim uređajima ili za ugradnju u složeniji sustav. Svaki stupac sastoji se od 24 kontakata koji su numerirani prema oznakama sa slike 17. Kontakti `j1_1` i `j1_24` povezani su s izlazom iz regulatora napajanja 3.3 V (V_{CCIO}), dok su kontakti `j2_1` i `j2_24` povezani na uzemljenje (GND). Preostala 44 kontakata povezana su s FPGA sklopom, od kojih je 15 signala povezano i s komponentama ugrađenima na razvojnu pločicu (LED indikatori, tipke i težinska otporna mreža), pa je mogućnost njihove primjene ograničena. Popis signala koji su povezani i na ugrađene komponente i na priključnice za proširenje dokumentiran je u datoteci .



Deklaracija signala u sučelju sklopa (VHDL):

Slika 17: priključnice za proširenje

```

port (
    j1: inout std_logic_vector(23 downto 2);
    j2: inout std_logic_vector(23 downto 2);
);

```

2.14 Definicije vanjskih signala

Datoteku koja određuje poveznice između logičkih naziva svih ulazno-izlaznih signala i pripadajućih fizičkih priključaka FPGA sklopa može se dohvatiti s web sjedišta <http://www.nxlab.fer.hr/dl/ulx2s.lpf>.

3 Instaliranje programske podrške

Programsko okruženje za sintezu digitalnih sklopova Lattice Diamond dostupno je za operacijske sustave Microsoft Windows i Linux, uz licencu za besplatno korištenje za rad s jednostavnijim FPGA i CPLD sklopovima. Uz verziju za Microsoftove operacijske sustave isporučuje se i alat za simulaciju digitalnih sklopova Aldec Active-HDL. Lattice Diamond za rad zahtijeva računalo s najmanje 2 GByte radne memorije (RAM). U nastavku je dan pregled najvažnijih koraka za instalaciju programskih alata potrebnih za rad s razvojnom pločicom na operacijske sustave Microsoft Windows.

3.1 Instaliranje programskog paketa Lattice Diamond

S web sjedišta tvrtke Lattice Semiconductor (<http://www.latticesemi.com/>) dohvatite instalacijski paket alata Diamond. Veličina instalacijskog paketa je cca. 1.9 GB. Prilikom pokretanja instalacijske .exe datoteku korisnik mora imati administratorske ovlasti. Programski paket se preporuča instalirati u pretpostavljeni (*default*) direktorij C:\lsc, jer su u starijim verzijama uočeni problemi u radu u slučaju odabira instalacije na alternativni disk ili direktorij.

Na operacijskom sustavu Windows 10 nužno je instalirati Lattice Diamond verzije 3.9 s pretpostavljenim postavkama odnosno u punom opsegu, dakle podršku za sve serije FPGA sklopova, simulacijski paket i upravljačke programe, jer se pokazalo da instaliranje uz isključenje pojedinih opcija rezultira neispravnim radom, odnosno nemogućnošću kasnijeg pokretanja alata Diamond. Potrebno je odabrati "**node locked**" model provjere licence.

Prilikom ispunjavanja zahtjeva za dodjelu besplatne licence potrebno slijediti upute s web sjedišta www.latticesemi.com o načinu i formatu prijave MAC adrese Ethernet sučelja Vašeg računala. Datoteku `license.dat` potrebno je spremati u direktorij `c:\lsc\diamond\3.9\license` ili `c:\lsc\diamond\3.9_x64\license`, zavisno od instalirane verzije alata Lattice Diamond.

Nakon prvog pokretanja, alat Diamond 3.9 ponudit će opciju dogradnje (*patch update*) koju **nije potrebno ni preporučeno instalirati**, budući da se ponuđenom dogradnjom proširuje podrška isključivo za novije FPGA sklopove proizvođača Lattice Semiconductor, dok se programski moduli za FPGA XP2 ugrađen na razvojnu pločicu ne mijenja.

3.2 Instaliranje USB pogonskih programa (*drivera*) za razvojnu pločicu

Instaliranje USB pogonskih programa (*drivera*) za razvojnu pločicu ULX2S odvija se automatski po priključenju pločice na USB priključnicu računala. Računalo mora imati pristup Internetu kako bi operacijski sustav mogao dohvatiti *drivera*.

Ako je operacijski sustav po prvom priključenju pločice na računalo automatski

pronašao pogonske programe (*drivere*) na pločici će biti upaljen LED indikator iznad USB priključnice. U suprotnom, *drivere* je potrebno ručno dohvatiti s web sjedišta proizvođača USB sučelja: <http://www.ftdichip.com/FTDrivers.htm> te ih instalirati.

3.3 Instaliranje programskog alata za programiranje pločice

Razvojna pločica ULX2S ima ugrađeno USB sučelje za programiranje koje nije podržano *driverima* za programatore proizvođača Lattice Semiconductor, zbog čega je nužno instalirati posebni upravljački programski alat. Program `ujprog.exe` potrebno je dohvatiti s web sjedišta <http://www.nxlab.fer.hr/dl/ujprog.exe> i spremiti ga u direktorij `C:\Windows`. Za pisanje po direktorijima operacijskog sustava korisnik treba imati administratorske ovlasti.

Program `ujprog.exe` može ispravno raditi samo ako su na operacijski sustav prethodno instalirani i USB pogonski programi (*driveri*) za razvojnu pločicu.

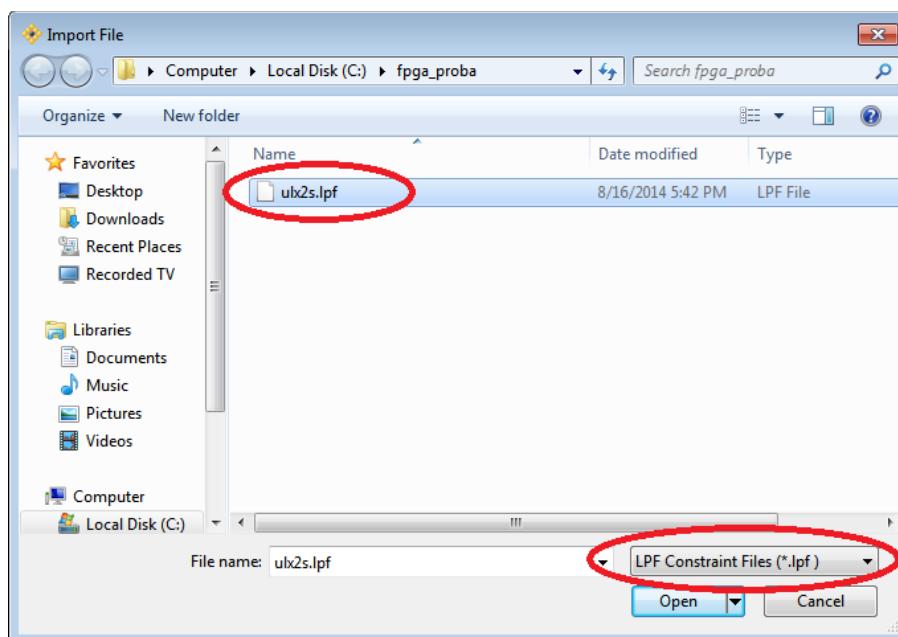
4 Rad u razvojnom okruženju Lattice Diamond

Glavna namjena razvojnog okruženja Lattice Diamond je sinteza konfiguracije FPGA sklopova iz odgovarajućeg opisa odnosno specifikacije digitalnog sklopa. Razvojno okruženje omogućuje opis digitalnog sklopa unosom shematskih dijagrama, ili korištenjem jezika za opis sklopova VHDL ili Verilog. U nastavku je dan primjer korištenja kroz nekoliko faza, od kreiranja projekta, unosa odnosno uređivanja shematskog opisa sklopa, sinteze konfiguracije te programiranja FPGA sklopa na razvojnoj pločici ULX2S, za što se koristi dodatni programski alat.

4.1 Stvaranje novog projekta

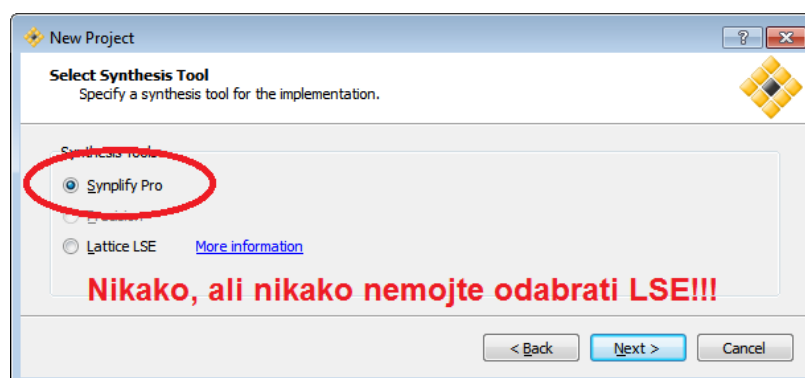
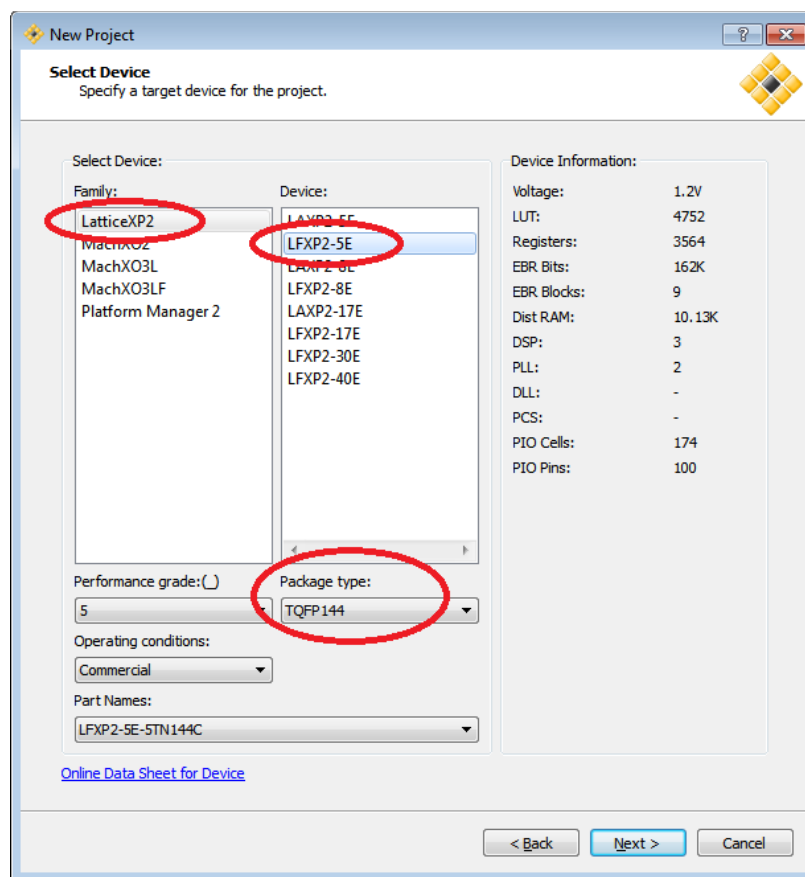
Pokrenite alat Lattice Diamond. Odaberite kreiranje novog projekta: *Project -> New*. Odaberite ime novog projekta, te radni direktorij u kojem će biti pohranjene ulazne datoteke vezane uz projekt.

U slijedećem koraku alat nudi izbornik za dodavanje već postojećih (unaprijed pripremljenih) datoteka u novi projekt. U projekt treba uključiti datoteku koja određuje poveznice između logičkih naziva ulazno-izlaznih signala (npr. LED_0) i fizičkih priključaka FPGA sklopa (npr. pin broj 29). Datoteku s definicijom svih logičkih signala može se dohvatiti s web sjedišta <http://www.nxlab.fer.hr/dl/ulx2s.lpf>.



U slijedećem koraku potrebno je odabrati točan tip FPGA sklopa. Na pločicu ULX2S ugrađuje se FPGA sklop **LFXP2-5E** ili **LFXP2-8E**, brzine rada (*speed grade*) 5, u kućištu **TQFP144**.

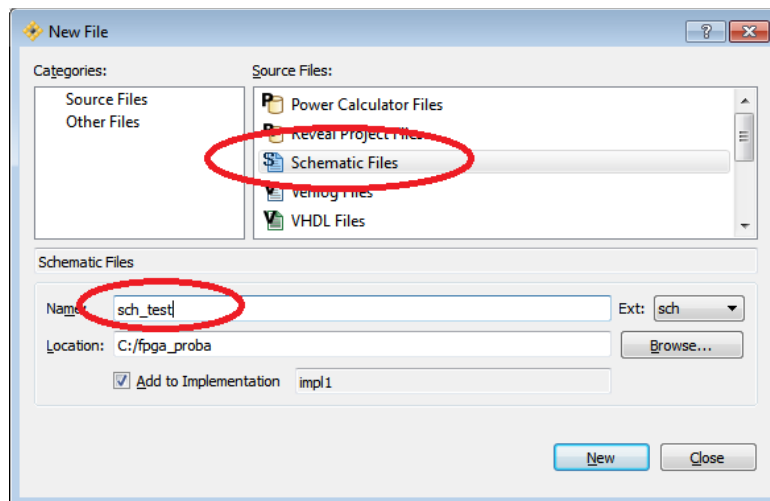
Nakon odabira FPGA sklopa otvara se izbornik u kojem se kao pretpostavljeni (*default*) alat za sintezu (*synthesis tool*) nudi *Lattice LSE*. U ovom koraku **obavezno** treba odabrati *Synplify Pro* umjesto *Lattice LSE*, jer *LSE* **ne generira ispravno** konfiguracije FPGA sklopova serije XP2!



Odabirom *Syntesis toola* završava postupak stvaranja novog Lattice Diamond projekta.

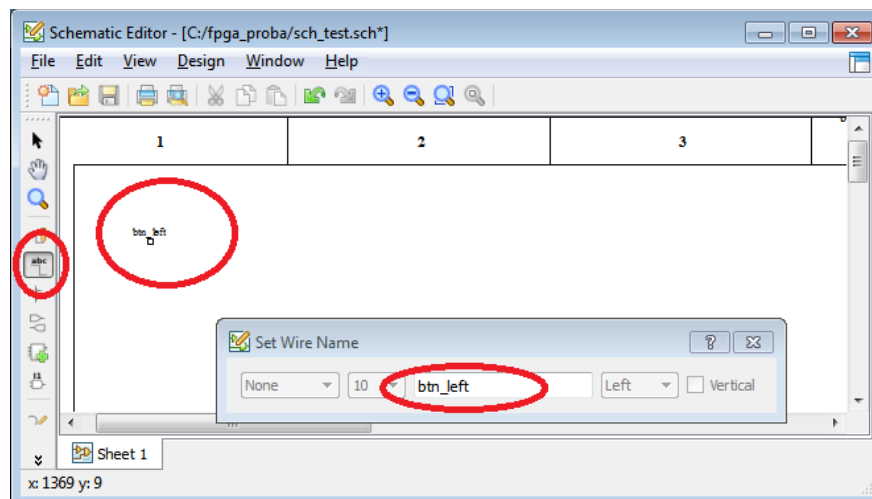
4.2 Shematski opis digitalnog sklopa

Kako bi mogli shematski opisati željeni digitalni sklop, potrebno je prvo stvoriti novu praznu *schematic file* datoteku i uključiti ju u trenutni projekt. Postupak za stvaranje novih VHDL ili Verilog datoteka i njihovo uključivanje u projekt je identičan, međutim zbog preglednosti i jednostavnosti u ovom primjeru prikazan je isključivo shematski način unosa odnosno opisa digitalnog sklopa. Otvaranje izbornika za dodavanje novih datoteka u projekt može se pokrenuti kroz izbornik *File – New – File*, ili pritiskom na tipke CTRL+N. U izborniku je potrebno odabrati "*Source files*" / "*Schematic files*", te datoteci dodijeliti proizvoljno ime (u ovom primjeru ime datoteke je `sch_test`).

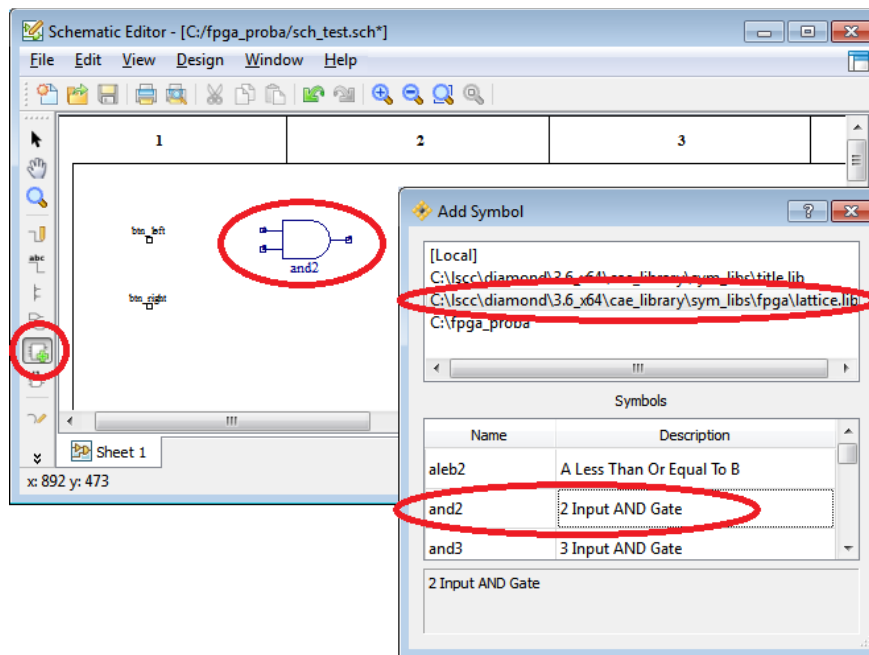


U ovom primjeru prikazan je postupak specificiranja sklopa s dva ulaza (tipke `btn_left` i `btn_right`) i dva izlaza (LED indikatori `led_0` i `led_1`), na koje se povezuju izlazi iz sklopova "I" (*and*) i "ILI" (*or*). Za lakši rad sa shematskim uređivačem (editorom) moguće je prozor uređivača izdvojiti iz integrirane radne površine programa Lattice Diamond desnim klikom miša na `tab sch_test.sch`, te odabirom opcije "Detach tool". Na taj način i na računalima s manjim zaslonom može se dobiti dovoljno velika radna površina za raspoređivanje elemenata u shematskom opisu sklopa. Prilikom uređivanja shematskog dijagrama korisno je poslužiti se opcijama "Zoom in" odnosno "Zoom out", kako bi komponente sklopa s kojim radimo bile prikazane uvećane na radnoj površini. Upravljanje povećanjem odnosno smanjenjem prikaza može se postići i pomoću *scroll* tipke miša uz istodobno pritisnutu tipku "CTRL".

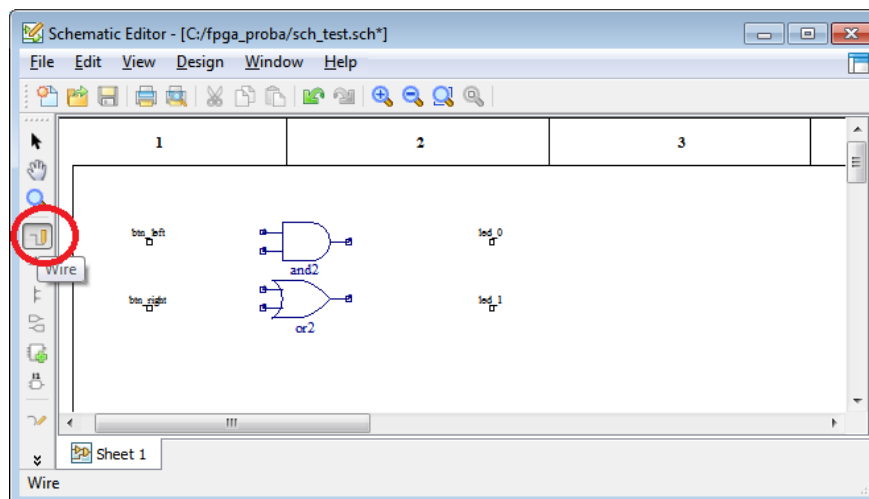
U prvom koraku treba stvoriti priključne točke za ulazne i izlazne signale, i dodjeliti im odgovarajuća imena. Dodjeljivanje imena signalima moguće je izvršiti odabirom alata "Net name" na izborničkoj traci s lijeve strane shematskog uređivača.



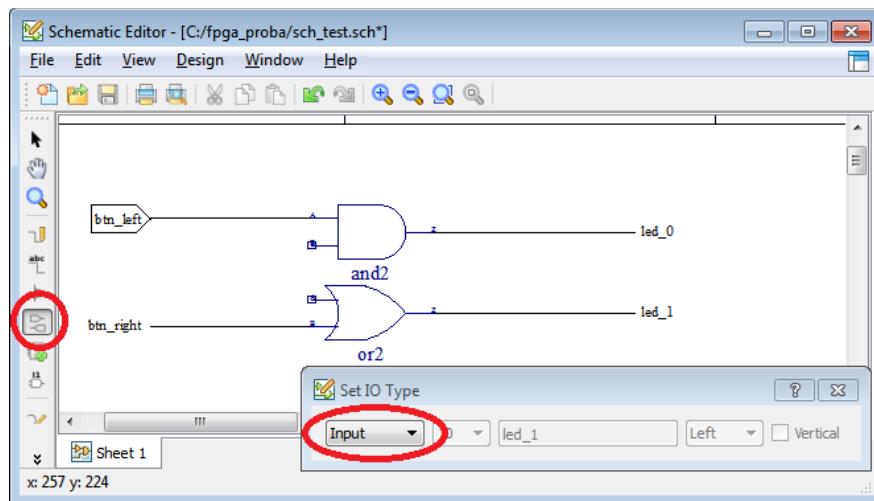
Nakon što su stvorene i ispravno imenovane oznake za sva četiri željena ulazno / izlazna signala, odabirom alata "Symbol" potrebno je otvoriti izbornik za postavljanje komponenti iz neke od dostupnih biblioteka u shematski dijagram. Treba odabrati biblioteku `lattice.lib`, iz koje se može u shematski dijagram uključiti po jednu instancu sklopova `and2` (dvoulazni sklop "I") i `or2` (dvoulazni sklop "ILI").



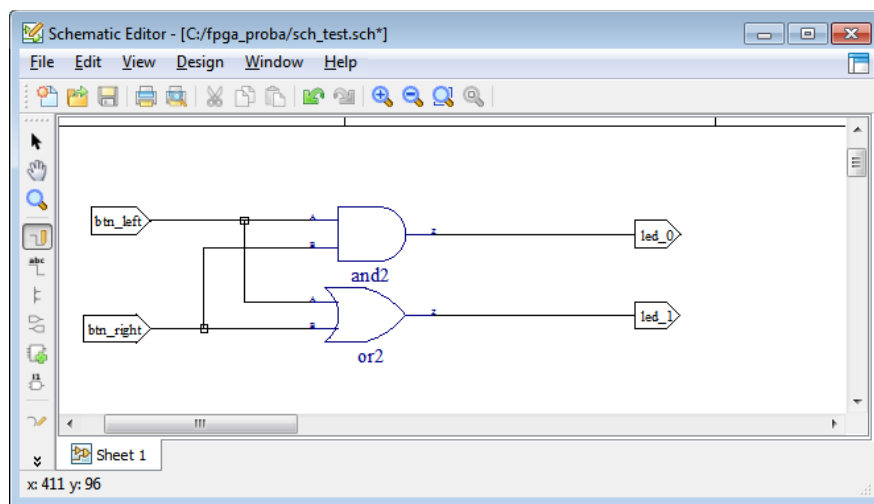
Nakon što su instance sklopova `and2` i `or2` postavljene na radnu plohu, instance sklopova `and2` i `or2` mogu se povezati s ulaznim odnosno izlaznim signalima odabirom alata "Wire" na izborničkoj traci s lijeve strane prozora.



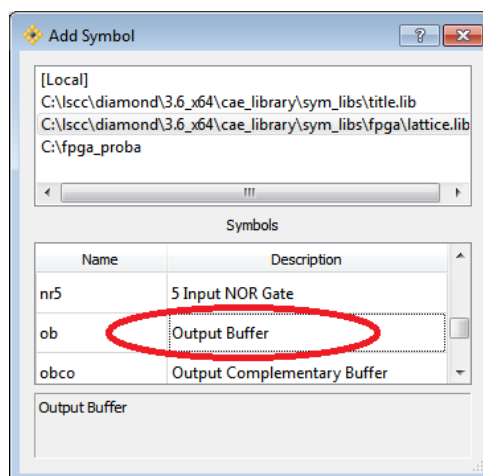
Nakon što su svi sklopovi i signali međusobno povezani na željeni način, potrebno je signale deklarirati kao ulazne, odnosno izlazne, korištenjem alata "IO Port" iz izborničke trake s lijeve strane prozora. U pomoćnom prozoru "Set IO Type" potrebno je odabrati ulazni (input) odnosno izlazni (output) tip signala, te na radnoj površini lijevim klikom miša pokazivačem odabrati (*select*) oznake imena željenih signala.

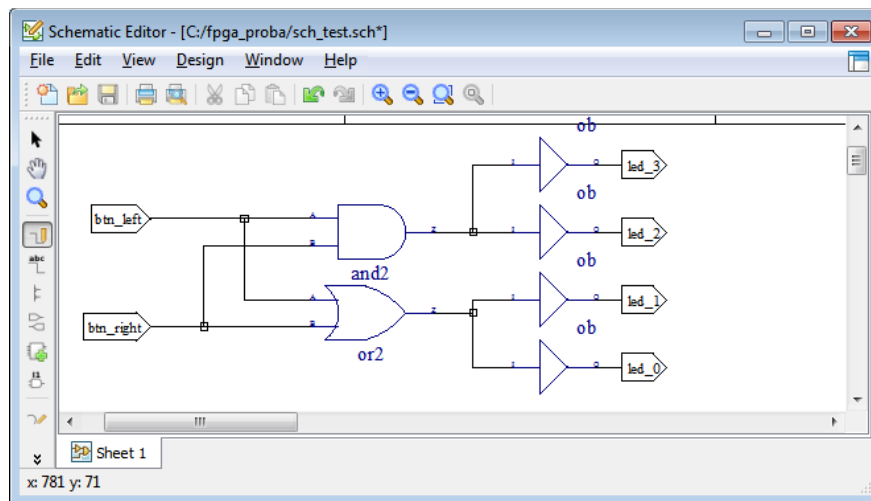


Ogledni sklop sad je potpuno specificiran i spreman za sintezu:



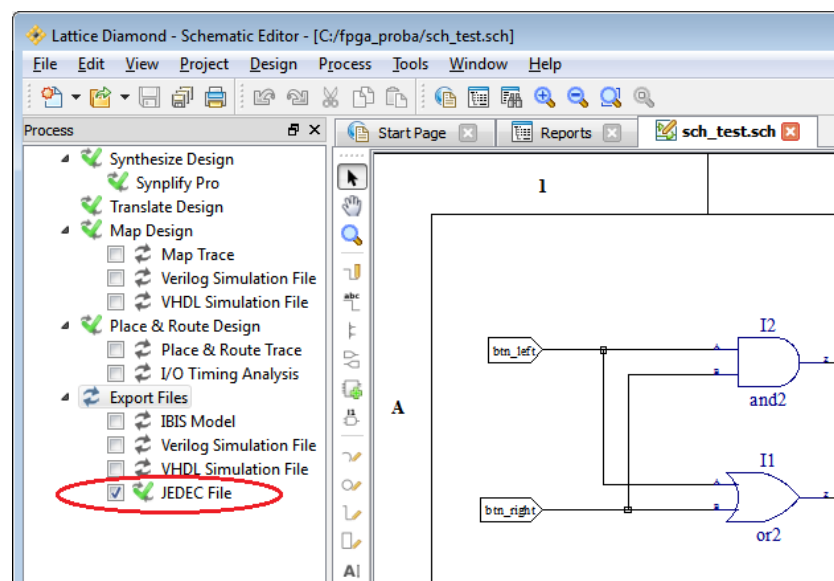
U slučaju da je određeni signal potrebno dovesti na više od jednog izlaznog sučelja, svakom izlazu treba dodijeliti zasebno izlazno pojačalo (*eng. output buffer*). U slijedećem primjeru svaki od izlaznih signala povezan je na zasebno izlazno pojačalo, koje se u biblioteci shematskih simbola krije pod skraćenicom *ob*.





4.3 Sinteza konfiguracije FPGA sklopa

Za pokretanje postupka sinteze konfiguracijskog *bitstreama* za odabrani FPGA sklop dovoljno je odabrati opciju "Export files" -> "JEDEC File" na kartici (tabu) "Process" na lijevoj strani prozora, te dvaput kliknuti mišem na opciju "Export files". Zavisno od brzine i opterećenja računala na kojem je pokrenuto programsko okruženje Lattice Diamond, proces sinteze konfiguracijskog *bitstreama* može potrajati od nekoliko desetaka sekundi do nekoliko minuta.



Uspješno proveden postupak sinteze alat Lattice Diamond označiti će "kvačicom" zelene boje uz svaki od provedenih koraka (*Synthesize Design*, *Map Design*, *Place & Route Design*, *Export Files*).

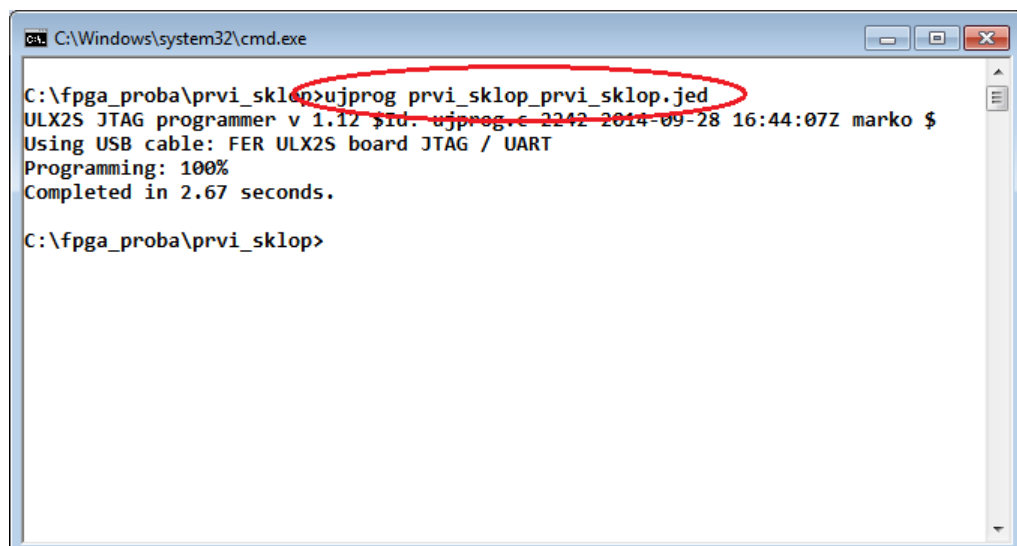
4.4 Programiranje FPGA sklopa

Za programiranje konfiguracijskog *bitstreama* na FPGA sklop razvojne pločice koristi se alat `ujprog`, koji je nezavisan od programskog okruženja Lattice Diamond, a poziva se iz komandne ljuske (*command shell*).

Konfiguracijsku datoteku za FPGA sklop treba potražiti u radnom direktoriju koji je korišten prilikom sinteze pomoću alata Lattice Diamond. U primjeru iz prethodnog poglavlja radni direktorij projekta je `C:\fpga_proba\prvi_sklop`. Konfiguracijske datoteke FPGA sklopova koje stvori alat Lattice Diamond imaju ekstenziju `.jed`. U ovom primjeru alat je stvorio konfiguracijsku datoteku pod imenom `prvi_sklop_prvi_sklop.jed`.

FPGA sklop može se direktno konfigurirati novom konfiguracijskom datotekom, čime konfiguracija FPGA sklopa postaje odmah aktivna, ali se gubi prekidom napajanja. Alternativno, konfiguracijska datoteka može se pohraniti u internu Flash memoriju FPGA sklopa, u kojem slučaju će ta konfiguracija automatski postati aktivna kod slijedećeg *reseta* FPGA sklopa.

Postupak direktnog konfiguriranja FPGA sklopa programom `ujprog` prikazan je slijedećom slikom. Za ispitivanje rada vlastitih sklopova **preporuča se koristiti ovu metodu programiranja FPGA sklopa**, jer je brza (programiranje traje samo nekoliko sekundi) i može se ponavljati proizvoljni broj puta bez utjecaja na trajnost FPGA sklopa. Konfiguracija FPGA sklopa programirana na ovaj način gubi se prekidom napajanja pločice. Tijekom programiranja LED indikator USB priključka na razvojnoj pločici treptati će u kratkim vremenskim intervalima.



```
C:\Windows\system32\cmd.exe
C:\fpga_proba\prvi_sklop>ujprog prvi_sklop_prvi_sklop.jed
ULX2S JTAG programmer v 1.12 $Id: ujprog.c 2242 2014-09-28 16:44:07Z marko $
Using USB cable: FER ULX2S board JTAG / UART
Programming: 100%
Completed in 2.67 seconds.

C:\fpga_proba\prvi_sklop>
```

Prije početka programiranja FPGA sklopa potrebno je zaustaviti sve programe koji komuniciraju s pločicom putem USB – RS-232 sučelja, npr. *Putty* ili *Hyper Terminal*.

Alternativno, konfiguracija FPGA sklopa može se pohraniti u njegovu internu *Flash* memoriju, čime se omogućuje da se FPGA sklop automatski sam konfigurira bez intervencije računala prilikom uspostave napajanja, što je nužno za primjene kad se razvojna pločica napaja iz baterijskog ili kakvog drugog izvora napajanja, nezavisno od računala. Međutim, u svakodnevnom radu **ne preporuča se korištenje ove metode**, jer je sporija, može se koristiti samo ograničeni broj puta (do 10.000 prema navodima proizvođača) te time utječe na trajnost FPGA sklopa, a u slučaju prekida napajanja ili komunikacije s razvojnom pločicom tijekom programiranja može doći i do trajnog oštećenja FPGA sklopa. Za programiranje interne konfiguracijske Flash memorije FPGA sklopa program `ujprog` je potrebno pozvati s uključenom opcijom `-j flash`, kako je prikazano na slici.

```
C:\Windows\system32\cmd.exe

C:\fpga_proba\prvi_sklop>ujprog prvi_sklop_prvi_sklop.jed
ULX2S JTAG programmer v 1.12 $Id: ujprog.c 2242 2014-09-28 16:44:07Z marko $
Using USB cable: FER ULX2S board JTAG / UART
Programming: 100%
Completed in 2.67 seconds.

C:\fpga_proba\prvi_sklop>ujprog -j flash prvi_sklop_prvi_sklop.jed
ULX2S JTAG programmer v 1.12 $Id: ujprog.c 2242 2014-09-28 16:44:07Z marko $
Using USB cable: FER ULX2S board JTAG / UART
Programming: 100%
Completed in 23.04 seconds.

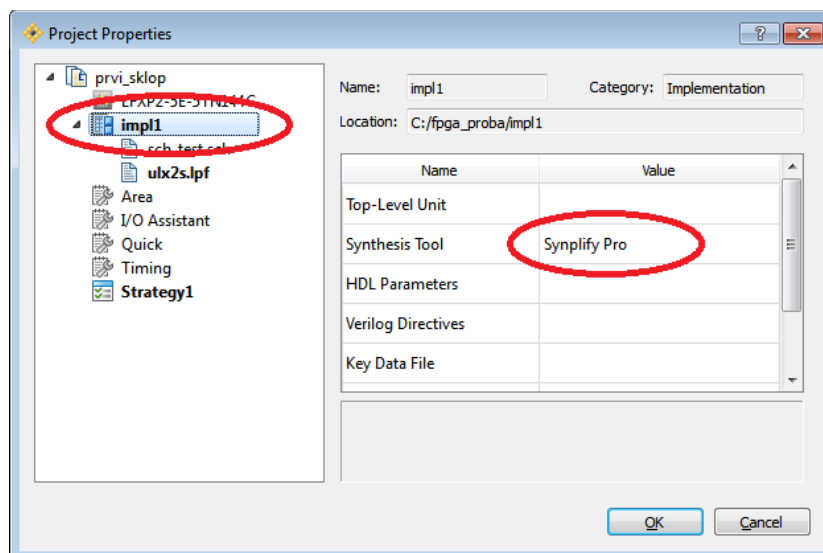
C:\fpga_proba\prvi_sklop>
```

Nakon što je FPGA sklop uspješno konfiguriran možemo ispitati njegov rad pritiskom na tipke `btn_left` i `btn_right`, što bi trebalo rezultirati paljenjem LED indikatora `led_0` i `led_1`, u skladu sa shematskom specifikacijom sklopa iz poglavlja 4.2.

4.5 Uobičajeni problemi u radu s FPGA razvojnim alatima

4.5.1 Nefunkcionalni alat za sintezu (*synthesis tool*)

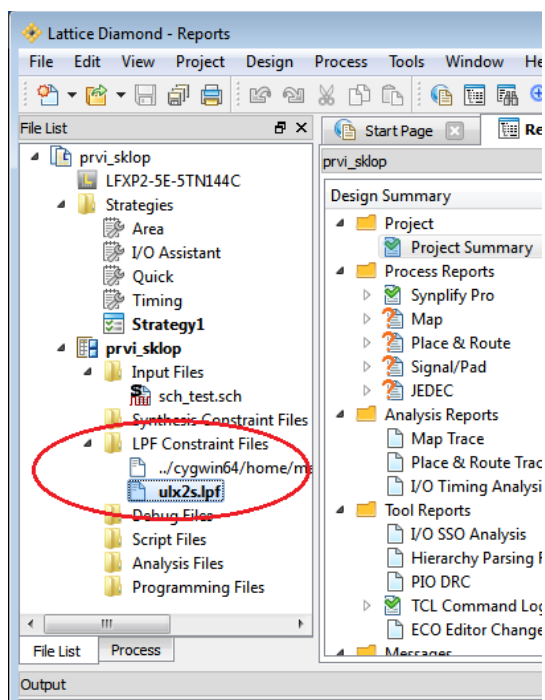
Ako je kod stvaranja projekta odabran alat za sintezu *Lattice LSE*, potrebno je koristeći izbornik *Project* → *Property Pages* odabrati *Synplify Pro*:



4.5.2 Definicije logičkih naziva vanjskih priključaka

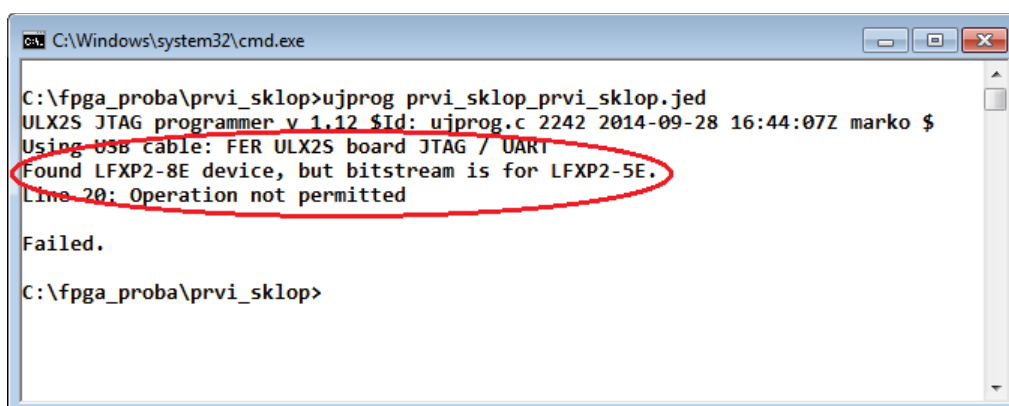
Ako u projekt nije uključena datoteka s definicijama vanjskih priključaka FPGA sklopa `ulx2s.lpf`, ili je ta datoteka prazna, sintetizator ne može povezati logičke nazive ulaznih i izlaznih signala s fizičkim pinovima. Tipični simptom je da sintetizirani sklop ne reagira na pobudu, a svi LED indikatora tinjaju slabim intenzitetom. U ovakvom slučaju u izborniku "File list" na lijevoj strani radne površine treba desnim klikom na ikonu "LPF

"Constraint Files" odabrati "Add existing file" te u projekt dodati datoteku `ulx2s.lpf`, nakon čega istu treba obavezno označiti desnim klikom miša odabirom opcije "Set as Active Preference File".



4.5.3 Odabir pogrešnog FPGA sklopa

Na pločicu ULX2S ugrađen je FPGA sklop **LFXP2-8E** ili **LFXP2-5E** u kućištu tipa **TQFP-144**. U slučaju pogrešnog odabira prilikom konfiguriranja FPGA sklopa javlja se poruka "Bitstream for unsupported target" ili "Found xxx device, but bitstream is for yyy". U izborniku "File list" potrebno je dvostrukim klikom na tip FPGA sklopa otvoriti odgovarajući prozor i odabrati ispravni tip sklopa.



4.5.4 Specijalni znakovi u imenu projekta i datoteka

Put do kazala (*foldera*) u kojeg su smještene datoteke projekta, kao i ime projekta te svih datoteka koje su u njega uključene, smiju sadržavati samo velika i mala slova engleske abecede, brojke, te znak "_". Korištenje ostalih znakova (posebno razmaka i zagrada, te hrvatskih dijakritičkih znakova) dovodi do pogrešaka u radu alata Lattice

Diamond.

4.5.5 Projekt bez datoteka

Datoteke koje nisu uključene u projekt mogu se uređivati (editirati) ali se ne mogu sintetizirati, pa se preporuča uređivati isključivo datoteke koje su vidljive u izborniku "File list" u odjeljku "Input files", na način da se u tom izborniku klikne na željenu datoteku.

4.5.6 Nepotrebne datoteke u projektu

Sličan problem predstavlja i uključivanje nepotrebnih datoteka u projekt, uslijed čega sintetizator ne može ispravno odabrati glavni (*top-level*) modul, ili prijavljuje različite poruke o grešci prilikom sinteze, ili sinteza rezultira nefunkcionalnim sklopom. U izborniku "File list" potrebno je iz popisa datoteka "Input files" izbrisati sve datoteke koje su eventualno greškom uključene u projekt.

4.5.7 Odabir glavnog (*top-level*) modula

Kod shematskog opisa sintetizator ponekad ne odabere ispravno glavni modul sklopa, što se događa u pravilu kad se prije prvog pokretanja postupka sinteze propusti u projekt uključiti sve potrebne datoteke, a tipično se radi o datotekama s definicijama shematskih simbola (.sym) koje povezuju implementaciju nekog od *black-box* modula s njegovim grafičkim prikazom u shematskom editoru. Problem se nakon dodavanja potrebnih odnosno uklanjanja suvišnih datoteka iz projekta može riješiti uključivanjem opcije "View – Show Views – Hierarchy" koja će indirektno rezultirati ispravnim odabirom glavnog (*top-level*) modula.

5 Opis sklopova jezikom VHDL

Kod kompleksnijih sklopova shematski opis postaje nepregledan, a uređivanje shema vremenski zahtjevno, iscrpljujuće te podložno čestim pogreškama koje je teško uočiti. Alternativu shematskom opisu predstavljaju jezici za opis sklopova (*hardware description languages*) od kojih su danas u praktičnoj upotrebi najviše zastupljeni Verilog i VHDL.

Od važnijih svojstva jezika VHDL mogu se istaknuti ova:

- Jezikom se prvenstveno opisuje struktura sklopa, na način da se *instanciranjem* manjih komponenti i njihovim međusobno povezivanjem oformljuje hijerarhijska struktura, od kojih isključivo modul na vrhu hijerarhije određuje povezivanje s fizičkim ulaznim odnosno izlaznim priključcima FPGA sklopa.
- Funkcioniranje pojedinih dijelova sklopa može se osim povezivanjem komponenti opisati i *ponašajno* - korištenjem logičkih ili aritmetičkih izraza te *ponašajnih blokova*, koje sintetizator preslikava u sklopovsku strukturu, a čime se često može pojednostavniti i poboljšati preglednost opisa sklopa.
- Sve instancirane komponente i ponašajno opisani blokovi preslikavaju se u nezavisne dijelove sklopovlja FPGA čipa, koji rade paralelno.
- Za opis standardnih tipova komponenti, npr. multipleksora, dekodera, komparatora, zbrajala, množila, bistabila, registara i memorija preporuča se koristiti (manje-više) standardizirane predloške koje sintetizator može prepoznati i na temelju takvih opisa optimalno odabrati i odgovarajuće povezati dijelove FPGA sklopovlja.

- Za razliku od shematskog opisa, gdje su poveznice (signali) između komponenti predstavljene crtama bez posebnih oznaka, svaka poveznica u VHDL-u mora imati jedinstveno ime i unaprijed deklariran tip, a vanjski priključci pojedinog modula moraju imati deklariran i smjer (ulaz, izlaz, ili dvosmjerna sabirnica).
- Vanjski priključci svakog modula deklariraju se u zasebnom dijelu, tzv. sučelju (blok *entity*), dok se interne poveznice i implementacija (struktura i ponašanje) sklopa opisuju u odvojenom bloku (*architecture*).
- Izvedeni tipovi podataka kao i sučelja unaprijed pripremljenih komponenata (npr. primitiva specifičnih za pojedini tip FPGA sklopa) deklariraju se bibliotekama (*library*) koje je potrebno uključiti u zaglavlje opisa modula.

Projekt u alatu Diamond sa sklopom opisanim VHDL-om može se stvoriti izvođenjem istih koraka kao i u poglavlju 4.1, nakon čega umjesto shematske treba kreirati novu datoteku s VHDL opisom: u izborniku *File – New – File* odabrati "*Source files*" / "*VHDL files*", te datoteci dodijeliti ime. VHDL opis sklopa koji je funkcijski i strukturno identičan shematski opisanom sklopu iz poglavlja 4.2 mogao bi izgledati ovako:

```

1  -- standardne biblioteke
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  -- biblioteke specifične za FPGA sklop Lattice XP2
6  library xp2;
7  use xp2.components.all;
8
9  -- deklaracija sučelja sklopa
10 entity sklop is
11 port (
12     btn_left, btn_right: in std_logic;
13     led_0, led_1: out std_logic
14 );
15 end sklop;
16
17 -- implementacija
18 architecture impl1 of sklop is
19 begin
20     sklop_i: and2
21     port map (a => btn_left, b => btn_right, z => led_0);
22
23     sklop_ili: or2
24     port map (a => btn_left, b => btn_right, z => led_1);
25 end impl1;

```

Slika 18: strukturni opis jednostavnog kombinacijskog sklopa

Proizvoljni komentari u VHDL-u započinju slijedom od dva znaka -- i nastavljaju se do kraja tekućeg retka (u primjeru sa slike 18 to su retci 1, 5, 9 i 17).

Biblioteka `IEEE.std_logic_1164` (uključena retcima 2 i 3 u ovom primjeru) definira tipove podataka i pripadajuće operatore za rad s višerazinskom logikom. Podatkovni tip `std_logic` definira ukupno devet razina signala te razrješenja konflikata pri povezivanju više razina signala na istoj sabirnici. Osim logičke nule ('0') i jedinice ('1'), pri opisu i sintezi sklopova važne su i razina visoke impedancije ('Z') koja se koristi za postavljanje priključka na sabirnicu u pasivno stanje, te proizvoljna razina (*don't care*, '-') koja omogućuje sintetizatoru efikasniju optimizaciju kombinacijske logike. Pri simulacijama su još posebno značajne neinicijalizirana ('U') i nedefinirana ('X') razina. Podatkovni tip `std_logic_vector` omogućuje grupiranje više bitova u

samo jedan naziv signala, pri čemu se svakom bitu može pristupiti i pojedinačno odgovarajućim indeksiranjem.

Jednobitne konstante (tipa `std_logic`) navode se u jednostrukim navodnicima, npr. '0', dok se višebitne konstante mogu navoditi u dvostrukim navodnicima u binarnom obliku, npr. "10101111", ili u heksadekadskom obliku uz prefix "x" ispred navodnika, npr. x"af".

Nad signalima tipa `std_logic` i `std_logic_vector` definirani su standardni operatori Booleove algebre: `not`, `and`, `or` te `xor`, `nand`, `nor` te `xnor`, pri čemu operator `not` ima viši prioritet od ostalih koji su svi jednakog prioriteta. Kod korištenja više različitih Booleovih operacija u jednom izrazu zato je potrebno koristiti zagrade kako bi se jasno specificirao redoslijed izvođenja operacija.

U primjeru sa slike 18 retcima 6 i 7 uključuju se deklaracije komponenti specifičnih za FPGA sklop Lattice XP2, od kojih se kasnije u implementaciji (od 20. do 24. retka) postavlja po jedna instanca modula `and2` i modula `or2`.

Sučelje sklopa deklarirano je od 10. do 15. retka u bloku `entity`. Unutar deklaracije `port()` nazivi signala istog tipa i smjera mogu se navoditi u jednom retku, a znakom ";" zaključuje se svaki redak deklaracije, osim zadnjeg. Nazivi signala moraju započinjati slovom engleske abecede, a mogu sadržavati i brojeve i znak "_". VHDL ne razlikuje velika i mala slova kako kod ključnih riječi tako ni kod naziva signala.

Implementacija sklopa opisuje se u bloku `architecture`. Interni tipovi podataka, signali i konstante deklariraju se na početku ovog bloka prije ključne riječi `begin`, ali u primjeru sa slike 18 nije bilo potrebe za deklaracijom internih signala ni konstanti. Korištenjem jezične konstrukcije `port map()` instanciraju se komponente iz vlastitih modula ili gotovih biblioteka. U primjeru je prikazan način povezivanja signala imenovanjem, pri čemu za svaki unutar izraza `port map()` navode parovi naziva signala povezani slijedom znakova "=>". S lijeve strane navodi se naziv signala iz sučelja komponente, a s desne strane naziv signala u trenutnom modulu. Alternativni način specificiranja povezivanja signala sa sučeljem komponente slijednim nabranjanjem samo signala trenutnog modula (tzv. pozicijsko povezivanje) kraći je, ali nepregledniji i podložniji pogreškama, pa se ne preporuča.

Kako se u primjeru sa slike 18 radi o jednostavnom kombinacijskom sklopu, umjesto instanciranja primitiva vezanih za specifičnu biblioteku, identična funkcionalnost može se postići korištenjem Booleovih operacija u tzv. konkurentnim izrazima, kao što je prikazano na slici 19:

```

Source Editor - [C:/fpga_proba/vhdl_test.vhd]
File Edit View Window Help
1  -- standardne biblioteke
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  -- deklaracija sucelja sklopa
6  entity sklop is
7  port (
8      btn_left, btn_right: in std_logic;
9      led_0, led_1: out std_logic
10 );
11 end sklop;
12
13 -- implementacija
14 architecture impl1 of sklop is
15 begin
16     led_0 <= btn_left and btn_right;
17     led_1 <= btn_left or btn_right;
18 end impl1;

```

Slika 19: izvedba kombinacijske logike konkurentnim izrazima

U retcima 16 i 17 uočite operator pridruživanja "<=>" koji signalu s lijeve strane dodijeljuje vrijednost izraza s desne strane. Ovakav opis sklopa kompaktniji je, pregledniji, lakši za održavanje, te nije ovisan o dostupnosti i specifičnostima nestandardnih biblioteka.

Isključivo za potrebe simulacijske analize moguće je specificirati proizvoljno vremensko kašnjenje od promjene stanja ulaznih signala konkurentnog izlaza do promjene rezultata izraza (ključna riječ AFTER). Međutim, prilikom sinteze konfiguracije FPGA sklopa takva umjetno zadana kašnjenja bit će zanemarena jer su stvarna kašnjenja uvjetovana fizičkim svojstvima sklopova i kod modernih FPGA platformi reda su veličine od nekoliko stotina ps do nekoliko desetaka ns, zavisno od kompleksnosti opisane kombinacijske logike i duljini odabranih prospojnih puteva unutar FPGA sklopa.

U nastavku poglavlja mogu se pronaći predlošci za opis nekih često korištenih kombinacijskih i sekvencijskih struktura.

5.1 Multipleksor

Primjer opisa multipleksora 4-u-1 s ulazom za omogućavanje:

```

library ieee;
use ieee.std_logic_1164.all;

entity mux4le is
port (
    d: in std_logic_vector(3 downto 0);
    a: in std_logic_vector(1 downto 0);
    e: in std_logic;
    y: out std_logic
);
end mux4le;

architecture impl1 of mux4le is
    signal y0: std_logic;
begin
    with a select y0 <=
        d(0) when "00",
        d(1) when "01",
        d(2) when "10",
        d(3) when others;
end impl1;

```

```
y <= y0 and e;  
end impl1;
```

5.2 Dekoder

Primjer opisa binarnog dekodera 2-u-4 s ulazom za omogućavanje:

```
library ieee;  
use ieee.std_logic_1164.all;  
  
entity dek24e is  
port (  
    a: in std_logic_vector(1 downto 0);  
    e: in std_logic;  
    y: out std_logic_vector(3 downto 0)  
);  
end dek24e;  
  
architecture impl1 of dek24e is  
    signal y0: std_logic_vector(3 downto 0);  
begin  
    with a select y0 <=  
        "0001" when "00",  
        "0010" when "01",  
        "0100" when "10",  
        "1000" when others;  
    y <= y0 when e = '1' else (others => '0'); -- ovo tu je zapravo komparator  
end impl1;
```

6 Razvojna pločica kao samostalni mikroprocesorski sustav

Sintezom odgovarajuće procesorske jezgre na FPGA sklopu razvojna pločica može raditi i kao samostalni mikroprocesorski sustav. Velik broj različitih procesorskih jezgri opisanih jezicima VHDL ili Verilog dostupan je besplatno na <http://www.opencores.org/>. Tvrtka Lattice Semiconductor razvila je i vlastite, također besplatne 8-bitne i 32-bitne procesorske jezgre (<http://www.latticesemi.com/mico32>) koje su prilagođene njihovim FPGA sklopovima i za koje su dostupni i integrirani razvojni alati.

Prilagodba i sinteza za rad na FPGA sklopu mikroprocesorskog sustava temeljenog na *soft-core* procesorskim jezgrama, a posebno povezivanje s vanjskim memorijama i uređajima, te razvoj odgovarajuće programske podrške može biti kompleksan i vremenski zahtjevan zadatak. Zbog toga je za pločicu ULX2S pripremljena već gotova (sintetizirana) mikroprocesorska konfiguracija koja omogućuje komunikaciju i programsko upravljanje sa svim sklopovima ugrađenima na razvojnu pločicu, a temelji se na 32-bitnoj RISC procesorskoj jezgri *f32c* (<http://github.com/f32c/f32c>) koja radi na taktu frekvencije 81.25 Mhz.

6.1 Programiranje u razvojnom okruženju Arduino

Arduino (<http://arduino.cc>) popularna je razvojna okolina prilagođena jednostavnom programiranju ugradbenih mikroprocesorskih modula. Razvojnu pločicu ULX2S može se koristiti i kao mikroprocesorski sustav koji može pokretati programe razvijene u alatu Arduino. Više informacija o ovakvom načinu rada možete pronaći na web sjedištu <http://www.nxlab.fer.hr/fpgarduino/>.

6.2 Programiranje pomoću GNU razvojnih alata (C / C++)

Razvojna pločica može se koristiti kao 32-bitni RISC mikroprocesorski sustav, uz pripremu prema uputama koje su dostupne na <http://www.nxlab.fer.hr/dl/firmware.html>.

Za razvoj programa jezicima C i C++ bez okoline Arduino potrebno je pripremiti i instalirati odgovarajuće GNU razvojne alate. Na računalima s Microsoftovim operacijskim sustavima GNU razvojni alati moraju se instalirati unutar okruženja *Cygwin*, za koje se instalacijski paket može dohvatiti sa stranice <http://cygwin.org>. Prilikom instaliranja potrebno je odabrati opcionalne pakete `devel/gcc-core`, `devel/make` i `utils/diffutils`.

Nakon što je na računalo uspješno instalirana *Cygwin* radna okolina, potrebno je otvoriti interaktivnu *cygwin* ljsku unutar koje se provodi postupak instaliranja GNU razvojnih alata. U *home* direktorij potrebno je dohvatiti te raspakirati paket biblioteka i programa koje su dostupne na <https://github.com/f32c/f32c/archive/master.zip>. Slijedite upute za instaliranje GNU alata iz datoteke `f32c/src/compiler/patches/instalacija_gnu_toolova.txt`.

```
$ unzip master.zip
$ ls
f32c-master
$ less f32c-master/src/compiler/patches/instalacija_gnu_toolova.txt
```

Nakon što su GNU alati uspješno instalirani potrebno je podesiti *environment* varijablu `MAKEFILES` tako da pokazuje na datoteku `~/f32c-master/src/conf/f32c.mk`. U Bourne ljski (`bash`) koja je pretpostavljena ljska okruženja *Cygwin* to se postiže naredbom:

```
$ export MAKEFILES=~/f32c-master/src/conf/f32c.mk
```

Postavljanje varijable `MAKEFILES` može se automatizirati na način da se prethodna naredba doda u datoteku `~/ .bash_profile`.

Uz ispravno pripremljene GNU razvojne alate, prevođenjem programa u jeziku C najjednostavnije je upravljati putem već postojećih *make* pravila. U nastavku je prikazan primjer prevođenja oglednog C programa:

```
$ cd f32c/src/examples/
$ ls
hello_asm      hello_c        video_test
$ cd hello_c
$ ls
Makefile       hello.c
$ make
...
mips-elf-objcopy -O binary hello hello.bin
mips-elf-objcopy -O srec hello hello.hex
```

Prevedeni izvršni kod s ekstenzijom `.bin` može se kopirati na MicroSD karticu, te na FPGA pločici pokrenuti iz BASIC interpretera naredbom `exec`:

```
C:\Windows\system32\cmd.exe - uiprogram -t
ULX2S JTAG programmer v 1.12 $Id: uiprogram.c 2242 2014-09-28 16:44:07Z marko $
Using USB cable: FER ULX2S board JTAG / UART
Terminal emulation mode, using 115200 bauds
Press ENTER, ~, ? for help
Ucitavam sliku iz demo/pics/ivica.jpg
breaking at line 200
Ready
>dir "d:"
Directory for d:
  4194304  ulx2s_4m.img
           1712  hello.bin
4097 Kbytes in 2 files, 3874752 Kbytes free.
Ready
>exec "d:hello.bin"
Hello, FPGA world!
```

Ako se izvršna datoteka na MicroSD karticu pohrani pod imenom `bootme.bin`, kod uključanja pločice program će se učitati i pokrenuti automatski.