Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

# Digitalna logika

# Laboratorijske vježbe korištenjem sklopovskih pomagala

# Upute za 3. laboratorijsku vježbu

Marko Zec

Prosinac 2024.

# 1 Zadatak: opis kombinacijske logike jezikom VHDL

Vaš je zadatak jezikom VHDL opisati, sintetizirati za rad na FPGA sklopu, te ispitati kombinacijsku logiku kojoj je ulaz četverobitni binarni a izlaz osambitni ASCII kod. Izlazni kodovi trebaju odgovarati ASCII znakovima zadnjih osam znamenaka Vašeg JMBAG identifikatora, te sadržavati dodatne oznake kraja niza.

Kombinacijsku logiku treba ugraditi u unaprijed pripremljen kostur ispitnog sekvencijskog sklopa. Pomoćna logika ispitnog sklopa sastoji se od:

- četiri bridom okidana D bistabila (*D flip-flop*);
- zbrajala, koje trenutnu vrijednost binarnog koda spremljenog u bistabilima kombinacijskom logikom uvećava za jedan;
- okidačkih modula, koji pritiske na tipke razvojne pločice pretvaraju u impulse duljine točno jednog ciklusa takta, kojima se omogućuje upis novih vrijednosti u D bistabile, odnosno njihov reset;
- izlaznog modula, koji 8-bitni ASCII kod iz kombinacijske logike odašilje kao niz (seriju) bitova brzinom od 115200 bit/s u formatu pogodnom za prijenos putem USB sučelja na računalo i ispis ASCII znakova na zaslonu.

Svi sekvencijski moduli ispitnog sklopa okidaju se na rastući brid takta iz oscilatora ugrađenog na razvojnu pločicu (25 MHz). Kako bi se omogućilo vizualno praćenje stanja pobude i odziva kombinacijske logike, ispitni sklop uključuje i prospoj izlaza iz D bistabila na četiri LED indikatora razvojne pločice, te prospoj četiri najniža bita ASCII koda na dodatna četiri LED indikatora.

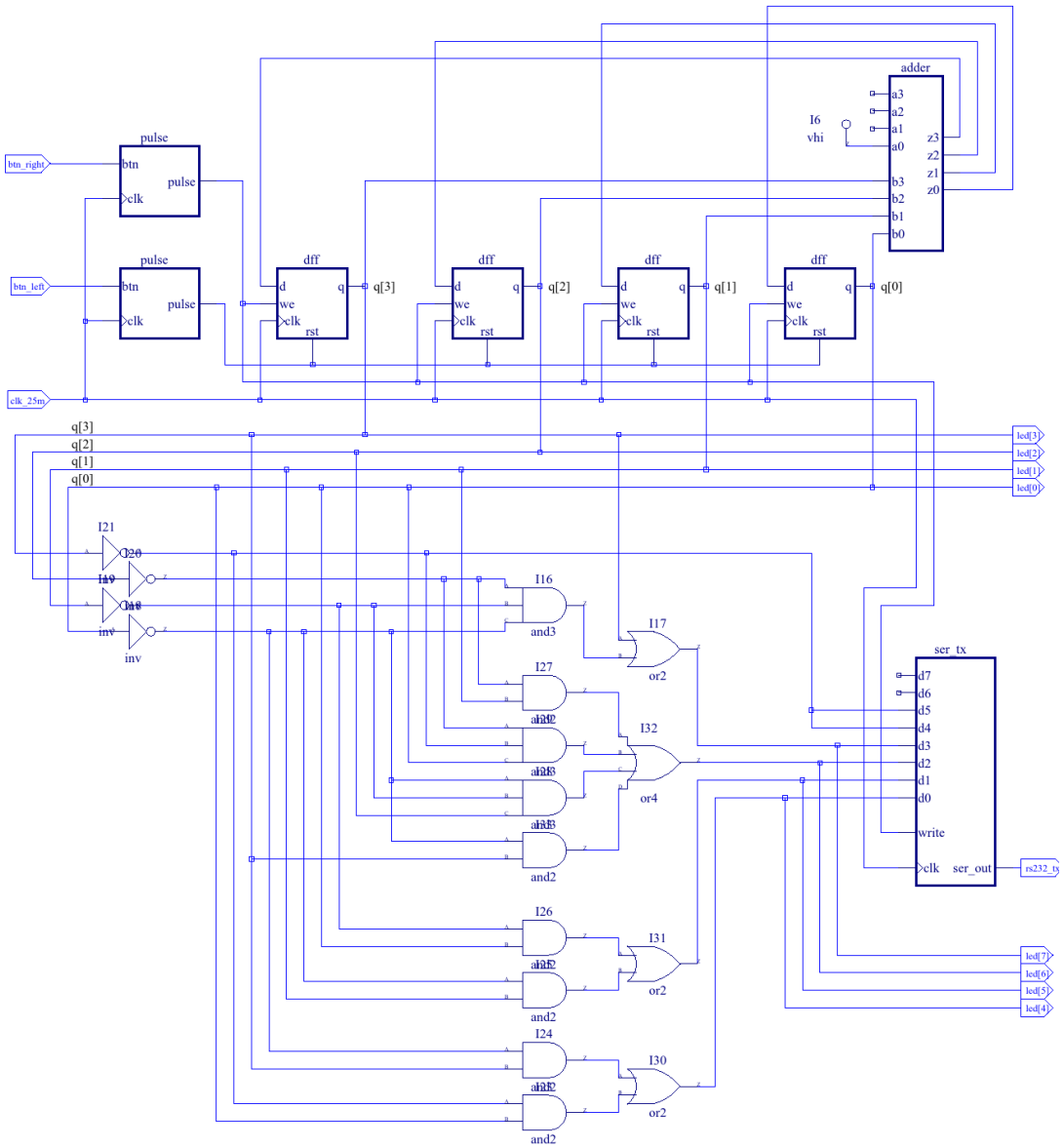
**Zadana funkcionalnost i struktura potpuno je identična sklopu koji ste već ostvarili shematskim opisom u prethodnoj (drugoj) laboratorijskoj vježbi**, a u drugom dijelu vježbe početni, najvećim dijelom strukturni opis, trebat će dodatno pojednostavniti zamjenom strukturnog povezivanja određenih modula funkcionalno ekvivalentnim ponašajnim opisom, odnosno direktnim izračunom vrijednosti signala putem tzv. konkurentnih izraza. Zbog toga nije potrebno ponovno crtati shemu, kao ni provoditi postupak minimiziranja kombinacijske logike, nego već gotovo rješenje zadatka iz prethodne laboratorijske vježbe koristite kao specifikaciju za ostvarenje ekvivalentnog sklopa jezikom VHDL.

Dakle, na laboratorijsku vježbu trebate doći samo s tablicom ASCII kodova JMBAG znamenki i oznaka za kraj retka, te s već gotovom (provjereno ispravnom odnosno ispitanom!) shemom sklopa iz 2. laboratorijske vježbe, bilo u digitalnom obliku ili na papiru.

Prije pristupanja ovoj laboratorijskoj vježbi proučite prva tri poglavlja iz skripte “Kratki uvod u jezik VHDL” koja je dostupna u repozitoriju predmeta na sustavu Ferko.

# 2 Strukturno i ponašajno opisivanje sklopova VHDL-om

Primjer sheme sklopa iz prethodne vježbe za (fiktivni) JMBAG 0087654321:




Umjesto shemom, u ovoj vježbi funkcionalnost i strukturu sklopa opisat će se jezikom za opis digitalnog sklopovlja VHDL (*Very High-Speed Integrated* ***Circuit Description Language***). Početni predložak VHDL opisa sklopa ekvivalentnog gornjoj shemi dostupan je kao popratni materijal uz upute za vježbu.

Za razliku od logičkih shema gdje signali (prospojne žice) nisu nužno imenovani, u VHDL-u svakom signalu mora biti dodijeljeno ime koje mora biti jedinstveno unutar pojedinog modula. Za imenovanje signala mogu se koristiti slova engleske abecede, znamenke od `0` do `9`, te znak `'_'`. Početni znak mora biti slovo, a važno je napomenuti da VHDL pri imenovanju signala ne razlikuje velika od malih slova, te da za imena nije dopušteno koristiti rezervirane ključne riječi samog jezika.

Vanjske signale, koji mogu biti ulazni (`in`), izlazni (`out`), te dvosmjerni (`inout`), deklarira se u sučelju VHDL modula (blok `entity`). Unutarnje signale deklarira se u implementacijskom dijelu (blok `architecture`) bez određivanja smjera.

Prilikom deklariranja signala određuje se njihov tip. Više signala istog tipa i smjera mogu (ali ne moraju) biti obuhvaćeni zajedničkom deklaracijom, gdje se pri nabrajanju više imena signala međusobno odvaja zarezom. U deklaraciji se ispred imena internih signala dodaje ključna riječ `signal`. Niz imena signala zaključuje se znakom ”`:`”, nakon kojeg slijedi oznaka smjera (samo za signale sučelja), te oznaka tipa, a deklaracija se zaključuje znakom ”`;`”, osim kod deklaracije zadnjeg signala sučelja.

Uočite razlike u deklariranju vanjskih i unutarnjih signala krovnog modula `lab3`:

```
entity lab3 is
    port (
        -- Deklaracije vanjskih priključaka (sučelje)
        clk_25m: in std_logic;
        btn_left, btn_right: in std_logic;
        rs232_tx: out std_logic;
        led: out std_logic_vector(7 downto 0)
    );
end lab3;

architecture x of lab3 is
    -- Deklaracije unutarnjih signala
    signal pulse_left, pulse_right: std_logic;
    signal q: std_logic_vector(3 downto 0);
    signal sum: std_logic_vector(3 downto 0);
    signal ascii: std_logic_vector(7 downto 0);
begin
    ...
```

U primjerima na laboratorijskim vježbama najčešće će se koristiti tip `std_logic` za jednobitne te `std_logic_vector` za višebitne signale. `std_logic` je višerazinski tip signala, koji osim binarnih vrijednosti `'0'` i `'1'` dopušta rad s dodatnim razinama. Razina visoke impedancije (*High-Z*) omogućuje promjenu smjera signala na priključnicama sklopa, a označava se znakom `'Z'`. Razine `'U'` kao neinicijalizirana te `'X'` kao nedefinirana ključne pri simuliranju rada sklopova, npr. „sudar” logičke nule i jedinice iz dva izvora na istom signalu rezultirat će nedefiniranom razinom (`'X'`).

Umjesto strukturnim povezivanjem odgovarajućih komponenti (logičkih vrata I, ILI odnosno NE kao u shemi iz prethodne vježne), kombinacijske jednadžbe mogu se u VHDL-u zadavati i direktno ponašajnim opisom tzv. konkurentnim izrazima, što je najčešće kraće, jednostavnije i značajno preglednije. U našem primjeru pojedinačnih bitova ASCII kodne riječi, izrazi koji odgovaraju kombinacijskoj logici iz zadane sheme bili bi ovi:

```
ascii(7) <= '0';
ascii(6) <= '0';
ascii(5) <= not q(3);
ascii(4) <= not q(3);
ascii(3) <= q(3) or (not q(2) and not q(1) and not q(0));
ascii(2) <= (not q(2) and q(1)) or (not q(3) and not q(2) and q(0))
  or (q(2) and not q(1) and not q(0)) or (q(3) and not q(0));
ascii(1) <= (not q(1) and q(0)) or (q(1) and not q(0));
ascii(0) <= (q(3) and not q(0)) or (not q(3) and q(0));
```

S lijeve strane izraza imenuje se odredišni signal kojem se zadaje vrijednost, nakon čega slijedi operator zadavanja vrijednosti „<=”, te konačno izraz čiji rezultat mora biti istog tipa kao i odredišni signal. Završetak izraza označava se znakom „;”.

Svaki konkurentni izraz tijekom postupka sinteze preslikat će se u zasebni dio FPGA sklopa, pa redoslijed zadavanja takvih izraza nije bitan, kao što ni prilikom crtanja shema nije bitan redoslijed ucrtavanja simbola ni njihov raspored: bitno je kako su u konačnici elementi međusobno povezani. Ova semantika ne vrijedi unutar blokova `process()` koje ćemo zbog toga nastojati izbjegavati prilikom prvih susreta s VHDL-om.

U gornjem primjeru, uz višebitne signale `ascii` i `q` koji mijenjaju vrijednost kroz vrijeme, u konkurentnim izrazima uočljive su i konstante `'0'`. Jednobitne konstante u VHDL-u zadaju se u jednostrukim navodnicima, a višebitne u dvostrukim. Umjesto u binarnom, često može biti praktično i preglednije višebitne konstante zadavati u heksadekadskom obliku, što se označava znakom ”x” ispred dvostrukih navodnika, pa su tako npr. konstante `x"9a3"` i `"100110100011"` ekvivalentne.

Uočite važnost prioriteta operatora prilikom opisivanja logičkih izraza VHDL-om. Unarni operator `not` najvećeg je prioriteta a djeluje nad vrijednosti s njegove desne strane. Složene izraze nad kojima se želi primijeniti operator `not` mora se obuhvatiti zagradama.

Logički operatori `and` i `or` niže su razine prioriteta od operatora `not`. U VHDL-u su logički operatori `and`, `nand`, `nor`, `xor`, `xnor` na istoj razini prioriteta, pa se u složenim izrazima gdje se koristi više različitih logičkih operatora redoslijed njihovog izvođenja obavezno mora istaknuti zagradama.

Pristup pojedinim bitovima ili nizovima bitova višebitnih signala omogućen je indeksiranjem koje se zadaje cjelobrojnim izrazima u zagradama. Na primjer, izrazi:

```
ascii(7) <= '0';
ascii(6) <= '0';
```

ekvivalentni su izrazu:

```
ascii(7 downto 6) <= "00";
```

Isto tako, izrazi:

```
ascii(5) <= not q(3);
ascii(4) <= not q(3);
```

ekvivalentni su izrazu:

```
ascii(5 downto 4) <= not q(3) & not q(3);
```

Uočite da operator „&” ovdje označava povezivanje više signala u jedan, a ne logičku operaciju. Dopušteno je i kombiniranje konstanti i signala u istom izrazu, npr:

```
ascii(7 downto 4) <= "00" & not q(3) & not q(3);
```

## 2.1 Predložak VHDL opisa, sinteza, te početno ispitivanje rada sklopa

Stvorite novi prazni direktorij na disku, te u njega raspakirajte arhivu `lab3.zip` koju možete dohvatiti s www.nxlab.fer.hr/dl.

Pokrenite razvojnu okolinu Lattice Diamond, te otvorite projekt `lab3_ulx2s.ldf` ili `lab3_ulx3s.ldf`, zavisno od razvojne pločice s kojom radite. Provjerite odgovara li tip FPGA sklopa onom koji je ugrađen na Vašu razvojnu pločicu, te prema potrebi u kartici "*File*" dvostrukim klikom na tip FPGA sklopa putem iskočnog izbornika odaberite ispravni.

U postavkama alata Lattice Diamond "*Tools, Options, Source Editor*" podesite parametar "*Tab stop*" na 8, kako bi VHDL opisi iz primjera bili ispravno formatirani.

Prije nego što počnete uređivati VHDL opis, preporučuje se sintetizirati ogledni sklop iz raspakirane arhive bez ikakvih promjena, isprogramirati FPGA sklop dobivenom konfiguracijskom datotekom, te ispitati rad sklopa.

Otvorite datoteku `lab3.vhd` kojom se instanciraju i povezuju moduli ispitnog sklopa, te pri samom kraju modula (od 112. retka) konkurentnim izrazima određuju funkcije bitova ASCII kodne riječi.

**Vaš je zadatak zamijeniti ogledne izraze onima koji ostvaruju ASCII kodne riječi koje odgovaraju Vašem JMBAG identifikatoru.**

Sintetizirajte sklop i konfigurirajte FPGA korištenjem alata `ujprog`. Tipkom `btn_right` okida se upis nove vrijednosti u bistabile, a tipkom `btn_left` stanje svih bistabila postavlja se na nulu. Ispitajte rad sintetiziranog sklopa opažanjem stanja LED indikatora i praćenjem odziva na računalu na isti način kao i na 2. laboratorijskoj vježbi.

## 2.2 Zamjena dijelova strukturnog opisa ponašajnim: zbrajalo

Opis krovnog ispitnog sklopa `lab3.vhd` instancira četiri razna tipa podređenih modula: `dff`, `adder`, `pulse` te `serial_tx`.

Pogledajmo opis modula `adder` u datoteci `adder.vhd`:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity adder is
    port (
        a3, a2, a1, a0: in std_logic;
        b3, b2, b1, b0: in std_logic;
        z3, z2, z1, z0: out std_logic
    );
end adder;

architecture x of adder is
    signal a, b, z: std_logic_vector(3 downto 0);
begin
    a <= a3 & a2 & a1 & a0;
    b <= b3 & b2 & b1 & b0;

    z <= a + b;

    z3 <= z(3); z2 <= z(2); z1 <= z(1); z0 <= z(0);
end x;
```

Uočite da je operacija zbrajanja određena samo jednim konkurentnim izrazom u jednom retku implementacijskog bloka `architecture`, dok interni signali `a` i `b` služe isključivo za prepakiravanje jednobitnih ulaza u višebitne argumente izraza kojim se provodi zbrajanje. Višebitni interni rezultat zbrajanja `z` raspakirava se u pojedninačne

jednobitne izlaze `z0`, `z1`, `z2` i `z3`. Ovakav oblik vanjskog sučelja modula `adder` bio je praktičan prilikom crtanja shematskog opisa korištenjem jednobitnih „žica” u prethodnoj vježbi, ali nije nužno i optimalan kad se za opis cijelog sklopa koristimo isključivo VHDL-om.

Pogledajmo i primjer instanciranja zbrajala `adder` u krovnom modulu `lab3`:

```
I_add: entity work.adder
port map (
    a3 => '0', a2 => '0', a1 => '0', a0 => '1',
    b3 => q(3), b2 => q(2), b1 => q(1), b0 => q(0),
    z3 => sum(3), z2 => sum(2), z1 => sum(1), z0 => sum(0)
);
```

Prilikom instanciranja modula prvo se zadaje labela odnosno proizvoljno odabrano jedinstveno ime. Labela u izvješćima alata za sintezu omogućuje razlikovanje više instanci identičnih komponenti unutar jednog modula.

Kod tzv. direktnog instanciranja slijedi ključna riječ `entity`, nakon koje se imenuje biblioteka i modul koji se instancira. Moduli koje u projektu definira sam korisnik implicitno se smatraju dijelom radne biblioteke `work`.

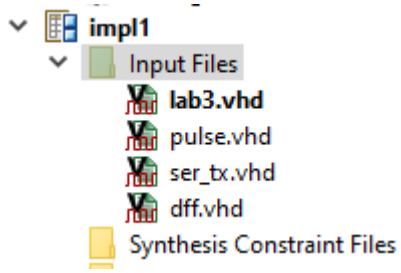
Povezivanje sučelja instance modula s lokalnim signalima zadaje se u bloku `port map`. S lijeve strane imenuje se određeni priključak sučelja instanciranog modula, nakon čega slijedi operator povezivanja ”`=>`”, a s desne strane operatora zadaje se naziv lokalnog signala koji se povezuje s odabranim priključkom. Parovi veza između priključaka instanciranog modula i lokalnih signala odvajaju se zarezom. VHDL inzistira da iza zadnjeg para ne smije biti zarez. Uočite da promatrajući samo instanciranje ne možemo odrediti koji priključci modula su ulazi, a koji izlazi, već je smjer određen deklaracijama u sučelju modula `adder`.

Opisani način povezivanja sučelja instanciranog modula s lokalnim signalima uobičajeno se naziva povezivanje po imenu. VHDL dopušta i skraćeni, pozicijski način notacije povezivanja, pri čemu se ispušta naziv priključaka sučelja instanciranog modula, a redoslijed zadavanja lokalnih signala u bloku `port map` mora odgovarati redoslijedu kojim su deklarirani priključci instanciranog modula. Pozicijski način povezivanja sučelja, koliko god izgledao primamljivo zbog kraćeg zapisa i sličnosti s notacijom zadavanja argumenata funkcijama u programskim jezicima, načelno treba strogo izbjegavati, jer bilo kakvom promjenom redoslijeda signala, kako u sučelju instanciranog modula, tako i na mjestu instanciranja, povezivanje može dati neželjene rezultate.

Može li se opis krovnog sklopa `lab3` pojednostavniti u dijelu koji uvećavanjem trenutne vrijednosti bistabila `q` za jedan određuje njihovo novo stanje na ulazima `d`?

Vaš je zadatak pojednostavniti opis krovnog sklopa `lab3` na način da **zamijenite instanciranje modula `adder` direktnim izračunom** vrijednosti signala `sum` odgovarajućim konkurentnim izrazom.

Kako se nakon dorada krovnog sklopa `lab3` više nigdje ne će instancirati modul `adder`, iz projekta je potrebno odjaviti njegov opis. U kartici „File List” alata Diamond klikom desnog gumba iznad datoteke `adder.vhd` na iskočnom izborniku odaberite „Remove”, nakon čega bi popis datoteka uključenih u projekt trebao izgedati ovako:



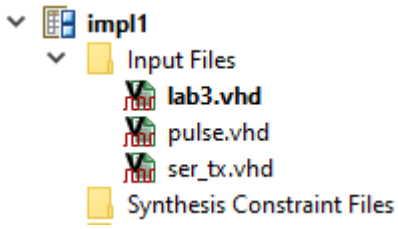

Sintetizirajte sklop, konfigurirajte FPGA korištenjem alata `ujprog`, te ispitajte rad sintetiziranog sklopa opažanjem stanja LED indikatora i praćenjem odziva na računalu.

## 2.3 Zamjena dijelova strukturnog opisa ponašajnim: bistabili

Može li se i funkcionalnost drugih instanciranih modula integrirati direktno u krovni modul `lab3` kao ponašajni opis, na sličan način kako ste u opisu krovnog modula direktno ostvarili funkciju izračuna nove vrijednosti koja se upisuje u bistabile?

Vaš je zadatak dodatno pojednostavniti opis krovnog sklopa `lab3` na način da **zamijenite instanciranje modula `dff` direktnim izračunom** vrijednosti signala `q` odgovarajućim konkurentnim izrazom. Kao predložak za opis bistabila iskoristite konkurentni izraz iz opisa modula `dff` u datoteci `dff.vhd`.

Kao i u prethodnom primjeru, nakon dorada krovnog modula, potrebno je iz projekta odjaviti modul `dff.vhd`, nakon čega bi popis datoteka uključenih u projekt trebao obuhvaćati samo `lab3.vhd`, `pulse.vhd` i `ser_tx.vhd`:




Sintetizirajte sklop, konfigurirajte FPGA korištenjem alata `ujprog`, te ispitajte rad sintetiziranog sklopa opažanjem stanja LED indikatora i praćenjem odziva na računalu.

## 2.4 Zamjena kombinacijskih jednadžbi multipleksorom i konstantama

VHDL omogućuje više načina opisivanja multipleksora. Primjer opisa multipleksora 16-u-1 s 8-bitnim podatkovnim ulazima i izlazom te 4-bitnim ulazom za odabir:

```
    with q select ascii <=
        x"41" when x"0", -- ASCII "A"
        x"42" when x"1", -- ASCII "B"
        x"43" when x"2", -- ASCII "C"
        x"44" when x"3", -- ASCII "D"
        x"0d" when x"4", -- ASCII <CR>
        x"0a" when x"5", -- ASCII <LF>
        x"00" when others;
```

U gornjem primjeru na ulaze multipleksora dovode se 8-bitne konstante, izlaz iz

multipleksora je signal `ascii`, a za odabir prospoja koristi se signal `q`. Za vrijednosti `q` koje nisu izrijekom specificirane izlaz će biti postavljen na vrijednost uz ključne riječi "`when others`", u konkretnom primjeru to je konstanta `x"00"`.

Vaš je zadatak u krovnom modulu `lab3` zamijeniti osam izraza za određivanje pojedinačnih bitova signala `ascii` samo jednim multipleksorom, koristeći gornji predložak.