Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Digitalna logika

Laboratorijske vježbe korištenjem sklopovskih pomagala

Upute za 3. laboratorijsku vježbu

Marko Zec

Studeni 2013.

1 Zadatak

Cilj vježbe je upoznavanje studenata s jezikom za opis digitalnih sklopova VHDL kroz praktični primjer projektiranja i ispitivanja rada jednostavnog kombinacijskog sklopa. U prvom koraku potrebno je korištenjem jezika VHDL opisati sklop koji je strukturno i funkcijski potpuno jednak sklopu kojeg je u 2. laboratorijskoj vježbi trebalo specificirati crtanjem sheme. U drugom dijelu vježbe sklop treba proširiti instanciranjem i međusobnim povezivanjem dodatnih modula.

1.1 Funkcije kombinacijskog sklopa (istovjetno 2. lab. vježbi)

Vaš je zadatak projektirati i ispitati kombinacijski modul "`enkoder`" koji će pritiske na tipke razvojne pločice (`btn_up`, `btn_down`, `btn_left`, `btn_right` i `btn_center`) preslikati u odgovarajuće 7-bitne kodove, na način da se pomoću 10 različitih ulaznih kombinacija pritisnutih tipki na izlazu sklopa dobije 10 različitih kodova prema unaprijed zadanoj tablici. Izlaze iz vlastitog kombinacijskog modula povežite na ulaz već gotovog modula za generiranje tonskog signala "`tonegen`", a izlaz iz generatora tonskog signala povežite na priključak za slušalice. Sve izlaze iz modula "`enkoder`" potrebno je dovesti i na LED indikatore, kako bi mogli provjeriti ispravnost generiranja zadanih 7-bitnih kodova. Ispravnost rada sklopa provjerite i priključenjem slušalica na pločicu te slušanjem tonova dobivenih kao odziv na kombinacije pritisnutih tipki.

Odabir kodnih riječi vrši se na temelju zadnje dvije znamenke studentovog JMBAG identifikatora. Varijabla "B" dobiva se prema formuli:

B = (JMBAG mod 50) + 32

U slučaju da za Vaš JMBAG vrijednost varijable "B" prema ovoj formuli iznosi 60, u vježbi treba koristiti vrijednost B = 31 za odabir kodnih riječi.

Studenti kojima su zadnje dvije znamenke JMBAG identifikatora u rasponu od 0 do 49 trebaju projektirati sklop za generiranje kodne riječi prema stupcu "dur" iz priložene tablice, dok ostali studenti trebaju generirati kodne riječi prema stupcu "mol".

kombinacija tipki	kodna riječ (dur) (JMBAG mod 100) < 50	kodna riječ (mol) (JMBAG mod 100) >= 50
	0	0
down	0	0
left	B + 0	B + 0
center	B + 2	B + 2
up	B + 4	B + 3
right	B + 5	B + 5
down + left	B + 7	B + 7
down + center	B + 9	B + 8
down + up	B + 11	B + 10
down + right	B + 12	B + 12

Za sve ostale moguće kombinacije tipki (one koje nisu obuhvaćene tablicom) rad kombinacijskog sklopa nije specificiran, te je dozvoljeno da kombinacijski sklop na izlazu generira bilo kakvu kodnu riječ. Po vlastitoj želji možete odabrati i drugačije kombinacije tipki od predloženih, u kojem slučaju te kombinacije morate ispravno dokumentirati, odnosno označiti u tablici u sklopu pripremnog dijela vježbe.

1.2 Priprema za vježbu: JMBAG = B = dur / mol

Popunite tablicu kombinacija ulaznih signala (tipki) i pripadajućih izlaznih kodnih riječi u dekadskom i binarnom zapisu.

Ulaz (tipke): down left center up right	Izlaz: kodna riječ (dekadski)	Izlaz: code[i] (binarno)						
		6	5	4	3	2	1	0

Na laboratorijsku vježbu trebate doći s ispravno popunjenom tablicom, te po mogućnosti s izvedbom sklopa koja odgovara funkcijskim zahtjevima prvog dijela zadatka. Ukoliko ste sačuvali pripremu s 2. laboratorijske vježbe, tablicu ne trebate ponovo popunjavati, nego je dovoljno donijeti pripremu s prethodne vježbe.

2 Opis digitalnih sklopova pomoću jezika VHDL

VHDL *(**V**ery high speed integrated circuits **h**ardware **d**escription **l**anguage)* je jezik za opis digitalnih sklopova i simulacijskih ispitnih okruženja za digitalne sklopove. U izvođenju ove vježbe koristiti će se podskup jezika VHDL koji omogućuje opis funkcionalnih digitalnih sklopova (tzv. *synthesizable VHDL*).

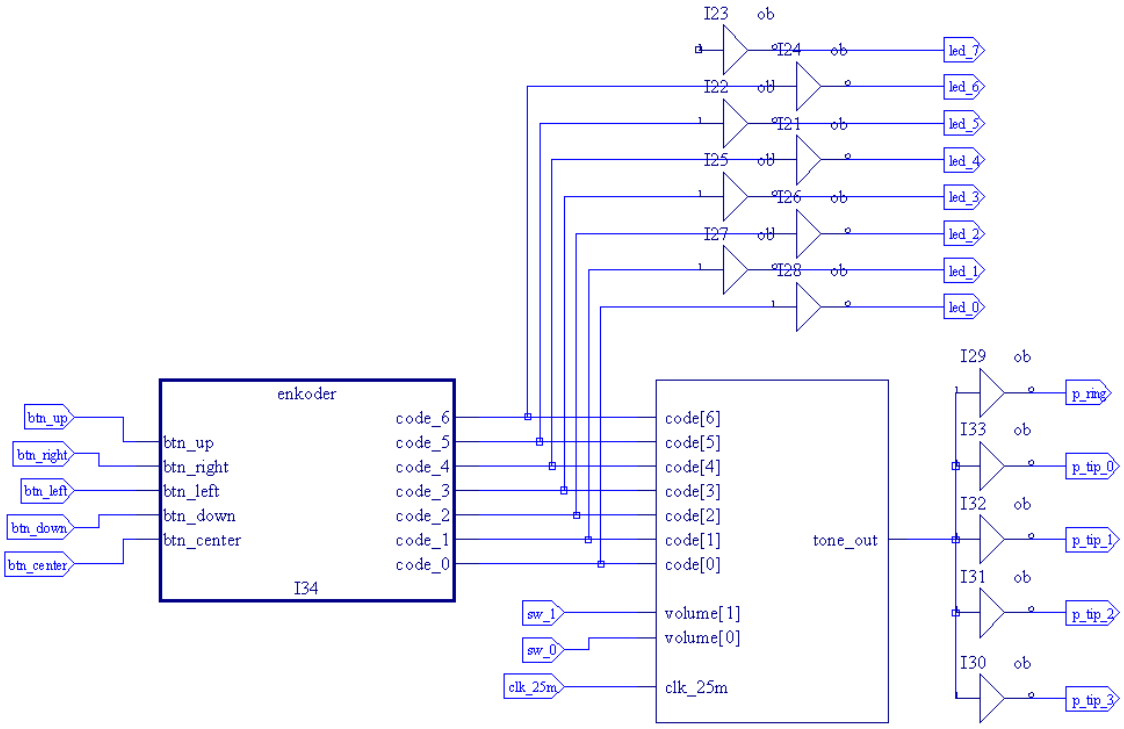
Prilikom upoznavanja sa strukturom, sintaksom i semantikom VHDL-a izrazito je bitno cijelo vrijeme imati u vidu da se **ne radi o programskom jeziku**, **već o jeziku kojim se opisuje struktura i ponašanje skupa međusobno povezanih digitalnih komponenti koje rade paralelno**, za razliku od programskih jezika kojima se formuliraju pravila za vremenski slijedno izvršavanje naredbi na računalu.

2.1 Sučelje sklopa: `entity`

Opis svakog sklopa (modula) u VHDL-u se sastoji od dva glavna bloka. Prvi blok, nazvan `entity`, opisuje vanjsko sučelje odnosno ulazne i izlazne signale modula. Drugi blok, nazvan `architecture`, opisuje unutarnju strukturu modula, odnosno

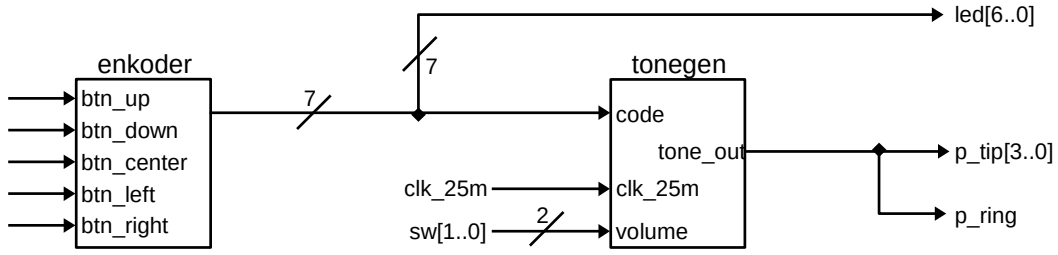
određuje njegovu funkcionalnost.

Prisjetimo se sheme sklopa "`sviraj`" iz 2. laboratorijske vježbe (slika 1). Glavne komponente sklopa su kombinacijski modul "`enkoder`" koji pritiske na tipke preslikava u 7-bitne MIDI kodove prema unaprijed zadanoj funkciji, te modul "`tonegen`" koji zavisno od 7-bitnog MIDI koda na ulazu generira odgovarajući tonfrekvencijski signal na izlazu. Izlaz iz sklopa tonegen dovodi se na oba kanala priključka za slušalice (`p_tip` i `p_ring`).



Slika 1. shematski opis sklopa "`sviraj`"

Isti sklop može se prikazati i alternativnom blok-shemom, u kojoj su višebitni signali prikazani samo jednom crtom (slika 2).



Slika 2. blok-shema sklopa "`sviraj`"

Sučelje modula "`enkoder`" iz ove sheme može se deklarirati na slijedeći način:

```
entity enkoder is
    port (
        btn_left, btn_right, btn_up, btn_down, btn_center: in std_logic;
        code: out std_logic_vector(6 downto 0)
    );
end enkoder;
```

Ovakvom deklaracijom definirano je ime modula (`enkoder`), te u bloku `port` nazivi (`btn_left`, `btn_right`, `btn_up`, `btn_down`, `btn_center`) i tip (`std_logic`) ulaznih signala (`in`), te naziv i tip izlaznog signala (`code`). Uočimo kako ovakva deklaracija sučelja, baš kao ni "crna kutija" koja simbolizira modul "`enkoder`" u shematskom prikazu, ni na koji način ne određuje njegovu funkciju ili ponašanje, već samo određuje na koji se način modul može povezivati s drugim komponentama.

Na sličan način može se deklarirati i sučelje glavnog modula (sklopa) "`sviraj`":

```
entity sviraj is
    port (
        clk_25m: in std_logic;
        sw: in std_logic_vector(3 downto 0);
        btn_left, btn_right, btn_up, btn_down, btn_center: in std_logic;
        led: out std_logic_vector(7 downto 0);
        p_tip: out std_logic_vector(3 downto 0);
        p_ring: out std_logic
    );
end sviraj;
```

Za razliku od modernih programskih jezika opće namjene (npr. C, C++, C#, Java), VHDL je *case-insensitive* jezik, što znači da će se rezervirane ključne riječi, te nazivi modula i signala tretirati jednako bez obzira na to jesu li pisani velikim ili malim slovima.

Uz ulazne i izlazne signale u bloku `port`, sučelje modula može (ali i ne mora) sadržavati i blok `generic`, putem kojeg se modulu mogu prenijeti proizvoljne konstante o čijoj vrijednosti može ovisiti njegovo ponašanje, npr:

```
entity enkoder is
    generic (
        B: integer
    );
    port (
        btn_left, btn_right, btn_up, btn_down, btn_center: in std_logic;
        code: out std_logic_vector(6 downto 0)
    );
end enkoder;
```

2.2 Najvažniji tipovi podataka

U opisu sučelja modula "`enkoder`" i "`sviraj`" kao tip ulaznih i izlaznih signala odabran je `std_logic` kod jednobitnih te `std_logic_vector` kod višebitnih signala. Tip podataka `std_logic` omogućuje modeliranje više vrijednosti jednobitnog signala, od kojih se pri opisu sklopova mogu koristiti:

- '`0`' .. logička nula
- '`1`' .. logička jedinica
- '`Z`' .. stanje visoke impedancije
- '`-`' .. bilo koja vrijednost (*don't care*)

Vrijednost '`Z`' (visoka impedancija) primjenjuje se kod opisa dvosmjernih signala odnosno sabirnica. U slučajevima kad za ispravan rad sklopa nije važno koju će vrijednost poprimiti određeni bitovi, vrijednost '`-`' može se koristiti kao uputa *(hint)* programskom alatu za sintezu kako bi optimalno minimizirao kombinacijsku logiku koja

proizlazi iz VHDL opisa (prisjetite se minimizacije modula "enkoder" korištenjem K-tablica u prethodnoj vježbi). Ostale vrijednosti koje može opisati tip podatka std_logic koriste se isključivo u simulacijama, zbog čega su izostavljene u ovom dokumentu.

Kako std_logic i std_logic_vector ne spadaju u temeljne tipove podataka ugrađene u VHDL, njihove definicije i operatori opisani su posebnim bibliotekama koje se moraju uključiti u svaki modul koji koristi navedene tipove podataka. Zaglavlje opisa modula "enkoder" s naredbama koje uključuju odgovarajuće biblioteke može izgledati ovako:

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

entity enkoder is
    generic (
        B: integer
    );
    port (
        btn_left, btn_right, btn_up, btn_down, btn_center: in std_logic;
        code: out std_logic_vector(6 downto 0)
    );
end enkoder;
```

Uz std_logic i std_logic_vector, tipovi podataka koji se često koriste u opisu sklopova su boolean i integer. Tip podatka *boolean* može poprimiti vrijednosti true i false, a najčešće se pojavljuje kao rezultat operacija usporedbe. Tip podataka integer ograničen je na raspon od – (2^{31} – 1) do + (2^{31} – 1), a glavna mu je prednost što je (ponekad) intuitivniji za prikaz cjelobrojnih vrijednosti od tipa std_logic_vector. Osim kao općenita zamjena za std_logic_vector, tip integer se često koristi pri opisu brojila, a jedini je tip podatka kojim se u VHDL-u mogu direktno indeksirati polja podataka (kompleksnije strukture kojima se obično opisuju ROM ili RAM memorije).

VHDL u pravilu ne dozvoljava implicitnu pretvorbu iz jednog tipa podataka u drugi, što ga svrstava u kategoriju tzv. s*trongly-typed* jezika. Za pretvorbu podataka potrebno je koristiti odgovarajuće funkcije iz standardnih biblioteka. Na primjer, slijedeća konstrukcija omogućuje pretvorbu podatka tipa integer u podatak tipa std_logic_vector:

```
-- Sve sto se napise desno od "--" smatra se proizvoljnim komentarom
-- I: integer;
-- S: std_logic_vector(15 downto 0);

S <= std_logic_vector(to_unsigned(I, 16));
```

Jedna od rijetkih ali u praksi vrlo korisnih iznimki od pravila zabrane implicitne pretvorbe tipa podataka su aritmetičke operacije zbrajanja i oduzimanja, kod kojih je dozvoljeno da jedan operand bude tipa std_logic_vector, a drugi tipa integer, dok je rezultat uvijek tipa std_logic_vector istih dimenzija kao i operand tipa std_logic_vector.

2.3 Implementacija sklopa: architecture

Unutarnja struktura sklopa (modula) odnosno njegova funkcionalnost opisuje se u bloku nazvanom architecture. Blok architecture u zaglavlju sadrži deklaracije svih

internih signala modula (dakle onih koji nisu vidljivi putem vanjskog sučelja), konstanti i tipova podataka, a u glavnom bloku sadrži niz konkurentnih naredbi ili blokova.

Konkurentnim naredbama ili blokovima opisuje se rad pojedinih komponenti sklopa. Komponente se mogu opisivati strukturno ili ponašajno. Strukturnim opisom instanciraju se već gotove komponente i povezuju s drugim komponentama ili blokovima putem internih signala, odnosno direktno prema vanjskim signalima deklariranima u sučelju sklopa (`entity`). Ponašajnim opisom zadaje se odziv komponente na pobudu, a alatu za sintezu se prepušta automatizirano preslikavanje funkcijske specifikacije u odgovarajuću mrežu elementarnih sklopova iz biblioteke specifične za ciljanu tehnologiju implementacije (CPLD, FPGA, ASIC...).

Ponašajni opis modula "`enkoder`" s parametrima B = 60; dur, bi mogao izgledati ovako:

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

entity enkoder is
    port (
        btn_left, btn_right, btn_up, btn_down, btn_center: in std_logic;
        code: out std_logic_vector(6 downto 0)
    );
end enkoder;

architecture behavioral of enkoder is
    -- interni signali se deklariraju ovdje, u zaglavlju bloka architecture
    signal btns: std_logic_vector(4 downto 0);

begin
    --
    -- Povezivanje pet odvojenih signala tipa std_logic
    -- u jedan signal tipa std_logic_vector sirine 5 bitova
    --
    btns <= btn_down & btn_left & btn_center & btn_up & btn_right;

    --
    -- Visebitni multipleksor 7 * 32 -> 7 s nepotpuno specificiranim izlazima
    -- Signal za odabir (select) je "btns"
    --
    with btns select
    code <=
        "0000000" when "00000", -- no tone
        "0000000" when "10000", -- no tone
        "0111100" when "01000", -- c -> MIDI #60
        "0111110" when "00100", -- d -> MIDI #62
        "1000000" when "00010", -- e -> MIDI #64
        "1000001" when "00001", -- f -> MIDI #65
        "1000011" when "11000", -- g -> MIDI #67
        "1000101" when "10100", -- a -> MIDI #69
        "1000111" when "10010", -- h -> MIDI #71
        "1001000" when "10001", -- c -> MIDI #72
        "-------" when others ; -- don't care
end;
```

3 Sinteza i ispitivanje rada sklopa

Prvi dio Vašeg zadatka je na temelju ovog predloška stvoriti vlastitu implementaciju modula "`enkoder`" koji odgovara parametrima B te dur/mol u skladu s Vašim JMBAG identifikatorom. Za povezivanje modula "`enkoder`" s modulom "`tonegen`" i vanjskim signalima na pločici (tipke, prekidači, LED indikatori, generator takta, priključak za slušalice) možete iskoristiti već gotovi predložak VHDL opisa sklopa "`sviraj`".

Usporedite kompleksnost rješenja u VHDL-u sa shematski opisanom implementacijom istog sklopa iz 2. laboratorijske vježbe.

3.1 Sinteza sklopa

Stvorite novi prazni direktorij na disku, te u njega pohranite slijedeće datoteke koje možete dohvatiti s web sjedišta laboratorijskih vježbi:

- **ulx2s.lpf** (definicije ulazno-izlaznih priključaka FPGA sklopa na pločici ULX2S)
- **sviraj.vhd** (VHDL opis sklopa koji povezuje module enkoder i tonegen)
- **enkoder.vhd** (VHDL opis pretvornika koda: tipke -> MIDI)
- **tonegen.vhd** (VHDL opis sinkronog sekvencijskog sklopa za generiranje tonskog signala)

Već gotove primjere modula "`sviraj`" i "`enkoder`" možete koristiti kao predloške za izradu vlastite vježbe. Drugim riječima, dozvoljeno ih je uz potrebne modifikacije uključiti u vlastiti projekt.

Pokrenite razvojnu okolinu Lattice Diamond, te stvorite novi projekt. Za radni direktorij projekta odaberite upravo stvoreni direktorij u kojem se nalaze datoteke dohvaćene iz repozitorija. FPGA sklop ugrađen na pločicu ULX2S je Lattice XP2-5E LFXP2-5E u TQFP-144 kučištu, o čemu treba voditi računa prilikom odabira ciljanog FPGA sklopa. Programirajte FPGA sklop sintetiziranom konfiguracijom te ispitajte njegov rad za svih 10 ulaznih kombinacija pritisnutih tipki promatranjem odziva na LED indikatorima, te slušanjem generiranih tonova na priključenim slušalicama.

3.2 Pitanja za provjeru znanja

- Proučite opis sklopa "`sviraj`". Čemu služi jezična konstrukcija "`port map`"?
- Čemu služi ključna riječ "`others`" koja se uspoređuje sa signalom za odabir ulaza multipleksora?
- Sintetizirajte i ispitajte rad sklopa za tri različite varijante izlaza multipleksora u sklopu "enkoder" za slučaj "`others`", prema predlošcima A, B i C. Za svaku od tri varijante zapišite opaženo ponašanje sklopa, te podatke o zauzeću LUT tablica (pogledajte izvješće "Process Reports -> Map", "Design summary" u alatu Lattice Diamond).

```
-- A:
"-------" when others;

-- B:
"0000000" when others;

-- C:
code when others;
```

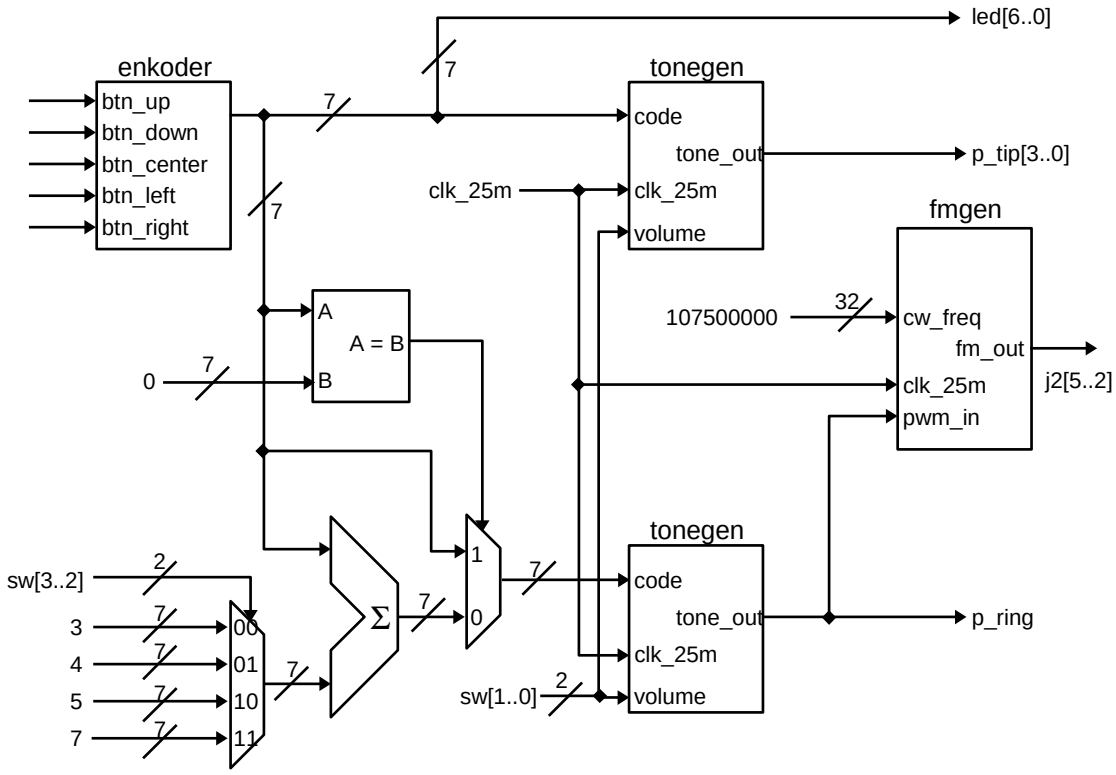
A) LUT4s:_____ Ponašanje sklopa:_____

B) LUT4s:_____ Ponašanje sklopa:_____

C) LUT4s:_____ Ponašanje sklopa:_____

3.3 Proširenje sklopa instanciranjem i povezivanjem dodatnih modula

U slijedećem koraku Vaš je zadatak proširiti sklop "`sviraj`" dodavanjem druge instance modula "`tonegen`" koji će generirati tonfrekvencijski signal za 3, 4, 5 ili 7 polustupnjeva kromatske ljestvice viši od signala koji generira prva instanca modula "`tonegen`". Odabir visine tona na izlazu drugog tonfrekvencijskog generatora vrši se prekidačima `sw(3)` i `sw(2)`. Ukoliko je svih sedam bitova na izlazu modula "`enkoder`" jednako nuli, kodna riječ na ulazu u drugu instancu modula "`tonegen`" također treba biti 0. Izlaze iz svake od dvije instance modula "`tonegen`" treba povezati odvojeno na lijevi, odnosno desni kanal priključka za slušalice (`p_tip` odnosno `p_ring`). Struktura modificiranog sklopa prikazana je blok-shemom na slici 3.



Slika 3. blok-shema sklopa "`sviraj`" proširenog dodatnim modulima

Dodatno, izlaz iz jedne instance modula "`tonegen`" (bilo koje) treba dovesti na ulaz `pwm_in` već gotovog modula za generiranje frekvencijski moduliranog radio signala "`fmgen`", kojeg možete preuzeti s web sjedišta laboratorijskih vježbi. Izlaz iz modula "`fmgen`" je radiofrekvencijski signal kojeg treba dovesti na nekoliko priključaka za proširenje razvojne pločice (npr. na priključke `j2[5..2]`), putem kojih će se signal emitirati u eter, a čak i bez priključenja dodatne žičane antene razina signala bi trebala biti dovoljna za detekciju i čistu demodulaciju na FM radijskom prijemniku ukoliko bude smješten u neposrednoj blizini razvojne pločice.

Na upravljački ulaz `cw_freq` modula "`fmgen`" trebate dovesti konstantnu 32-bitnu kodnu riječ koja određuje frekvenciju signala nositelja. Frekvencija signala nositelja treba biti u dijelu spektra koji se može primati običnim FM radio prijemnikom, odnosno u području od 88 do 108 MHz. Frekvencija se određuje prema formuli:

cw_freq = (88.0 + (JMBAG mod 100) / 5) * 1000000 Hz

Ispitajte rad sklopa za sve kombinacije prekidača `sw[3]` i `sw[2]` slušanjem tonfrekvencijskog signala na slušalicama priključenim na razvojnu pločicu, te prijemom radiofrekvencijskog signala na FM radio prijemniku. Zbog vrlo slabe razine radiofrekvencijskog signala koju pločica može emitirati udaljenost između razvojne pločice i radio prijemnika treba biti čim manja.

Važna napomena: multipleksore, komparator i zbrajalo ne treba izvoditi kao zasebne module, nego ih se može opisati korištenjem konkurentnih izraza `with-select` i `when-else`, odnosno operatora `+`, direktno u bloku `architecture` sklopa "`sviraj`". Primjer opisa zbrajala:

```
--
-- a i b moraju biti istog tipa i dimenzija
-- ako je c istog tipa kao a i b, c također mora imati iste dimenzija kao i a i b
-- alternativno, c može biti signal tipa integer ili cjelobrojna konstanta
--
a <= b + c;
```

Sve modifikacije izvedite u sklopu "`sviraj`". Modul "`enkoder`" ne treba modificirati!

Nakon što ste ispitali rad sklopa, u sustav Ferko trebate prenijeti (upload) slijedeće datoteke:

- **sviraj.vhd**
- **enkoder.vhd**
- **lab4.jed**

Ukoliko ste sklop projektirali i izveli koristeći drugačije ulazne kombinacije tipki od predloženih, obavezno trebate u sustav prenijeti i tekstualnu datoteku s popisom odabranih ulaznih kombinacija, u kojem slučaju dodatna datoteka treba biti nazvana

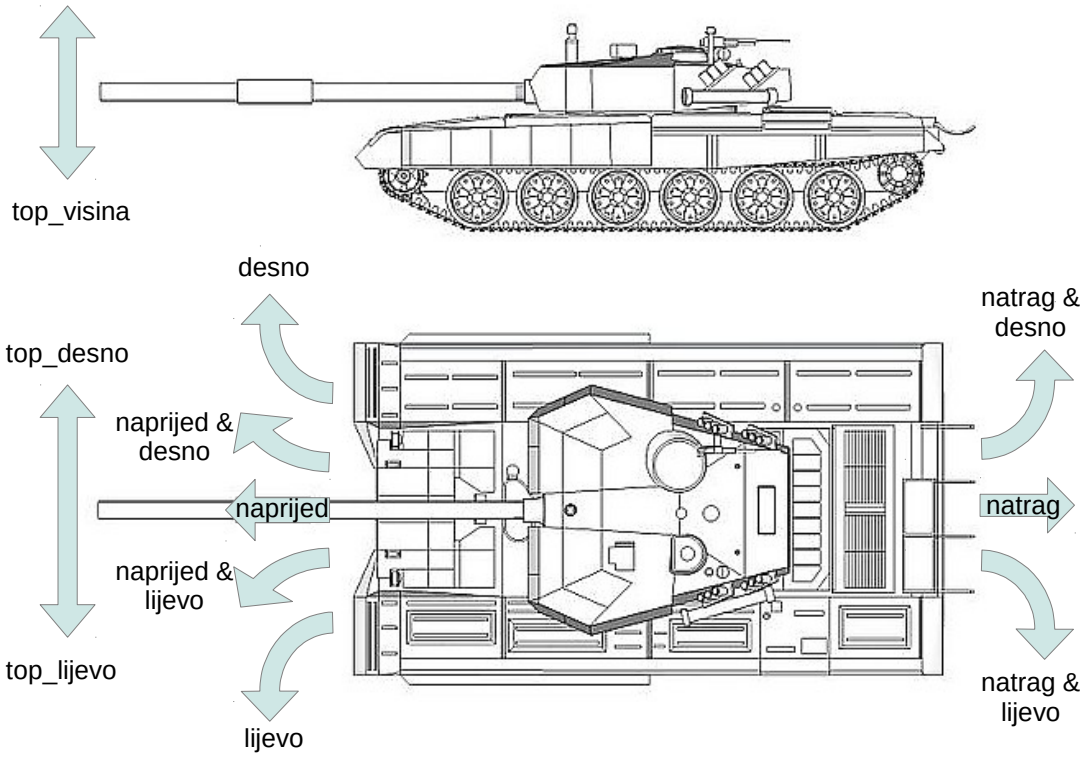
- **kombinacije.txt**

4 Dodatak: daljinski upravljač makete tenka

Važna napomena: ovaj dio vježbe nije obavezan i neće se ocjenjivati, a namijenjen je studentima koji kroz dodatni praktični primjer žele utvrditi razumijevanje temeljnih koncepata opisa digitalnih sklopova jezikom VHDL.

Vaš je zadatak konstruirati kombinacijski sklop koji će omogućiti upravljanje maketom tenka s većim brojem upravljačkih signala pomoću samo pet tipki na razvojnoj pločici ULX2S. Ciljani sklop je funkcijski potpuno identičan primjeru iz 1. laboratorijske vježbe, kojeg umjesto shematskim putem ovaj put treba opisati VHDL-om.

Pri izradi zadatka koristit ćete već gotovi modul "`rf_modulator`" koji ima slijedeće ulaze: `clk`, `kanal`, `brzina`, `naprijed`, `natrag`, `lijevo`, `desno`, `top_lijevo`, `top_desno`, `top_visina`, `top_zvuk`, `top_granata`, `strojnica` i `motor` (slika 4). Jedini izlaz iz modula "`rf_modulator`" je radiofrekvencijski signal `rf`, koji se može iskoristiti za prenošenje kodiranih upravljačkih informacija maketi tenka bežičnim putem na odabranom kanalu u frekvencijskom pojasu 27 MHz.



Slika 4: funkcije glavnih upravljačkih signala modula "`rf_modulator`"

Jezikom VHDL opišite sklop koji instancira modul "`rf_modulator`" te na njega dovedite sve signale potrebne za upravljanje maketom, a koji trebaju biti izvedeni kao logičke funkcije vanjskih ulaznih signala povezanih s pet tipki razvojne pločice: `btn_up`, `btn_down`, `btn_center`, `btn_left` i `btn_right`:

```
motor = btn_up AND btn_down AND NOT btn_center
naprijed = btn_up AND NOT btn_center
natrag = btn_down AND NOT btn_center
lijevo = btn_left AND NOT btn_center
desno = btn_right AND NOT btn_center
top_lijevo = btn_left AND btn_center
top_desno = btn_right AND btn_center
top_visina = btn_down AND btn_center
top_zvuk = btn_up AND btn_center
strojnica = btn_left AND btn_right
```

Izlazni signal `rf` modula "`rf_modulator`" treba povezati na priključnice višenamjenske stereo utičnice `p_tip` i `p_ring`, putem kojih se modulirani radiofrekvencijski signal odašilje u eter. Na dvobitni ulaz `kanal` modula "`rf_modulator`" treba dovesti vanjski signal s mikroprekidača `sw_1` i `sw_0`, a na ulaz `brzina` dovesti vanjski signal s mikroprekidača `sw_3` i `sw_2`.

Pri izradi zadatka možete se poslužiti predloškom za opis sklopa koji instancira i povezuje modul "`rf_modulator`" sa samo nekima od zadanih upravljačkih signala. Predložak je prikazan u okviru na slijedećoj stranici, a možete ga preuzeti i s web-sjedišta laboratorijskih vježbi.

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

entity daljinski_upravljac is
    port (
        clk_25m: in std_logic;
        sw: in std_logic_vector(3 downto 0);
        btn_left, btn_right, btn_up, btn_down, btn_center: in std_logic;
        p_tip: out std_logic_vector(3 downto 0);
        p_ring: out std_logic
    );
end daljinski_upravljac;


architecture behavioral of daljinski_upravljac is
    signal rf: std_logic;
    signal naprijed, natrag, motor: std_logic;

begin

    motor <= btn_up and btn_down and not btn_center;

    naprijed <= btn_up and not btn_center;
    natrag <= btn_down and not btn_center;

    odasiljac: entity rf_modulator
    port map (
        clk => clk_25m,
        kanal => sw(1 downto 0), brzina => sw(3 downto 2),
        naprijed => naprijed, natrag => natrag,
        lijevo => '0', desno => '0',
        top_lijevo => '0', top_desno => '0',
        top_zvuk => '0', top_visina => '0',
        top_granata => '0', strojnica => '0',
        motor => motor, rf => rf
    );

    -- Izlazni FM signal
    p_tip <= rf & rf & rf & rf;
    p_ring <= rf;
end;
```

Više detalja o funkcijama modula "`rf_modulator`", smjernice za ispitivanje rada sklopa za daljinsko upravljanje, te smjernice postupanje s radioupravljanim maketama možete pronaći u uputama za 1. laboratorijsku vježbu.