SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Projektiranje digitalnih sustava

laboratorijske vježbe
(VHDL)

Sadržaj

Vježba 1

Modeliranje jednostavnih komponenata

1.1 Radna okolina

Na svim računalima omogućen je rad pod korisničkim imenom `student` uz lozinku `student`. Studenti na računalima `korana` i `wilibald` VHDL simulator izvode lokalno, dok studenti na računalu `rorimac` VHDL izvode na računalu `wilibald` korištenjem protokola `telnet`. Svaka grupa treba na računalu na kojem koristi VHDL napraviti radni direktorij (`mkdir`) koji se treba nazvati prema imenu jednog od studenata iz grupe.

Za svaki zadatak potrebno je u radnom direktoriju grupe napraviti zaseban direktorij, u kojem će biti kreirane jedna ili više datoteka koje čine model te datoteka kojom se ispituje funkcionalnost modela. Tipično, svaka komponenta ili entitet(`entity`) se sprema u zasebnu datoteku sa `vhd` ekstenzijom. Datoteke modela se mogu unositi jednim od postojećih tekst editora `vi`, `nedit` ili `nano`.

Neka se rješenje zadatka sastoji od datoteka `modelBottom_1.vhdl`, `modelBottom_2.vhdl` i `modelTop.vhdl`. Tada se prevođenje modela obavlja naredbom:

```
$ vhdlp modelBottom_1.vhdl modelBottom_2.vhdl modelTop.vhdl
```

Valja obratiti pažnju da je redoslijed datoteka u komandnoj liniji bitan, tj. da datoteke moraju biti poredane od hijerarhijski nižih prema hijerarhijski višim komponentama.

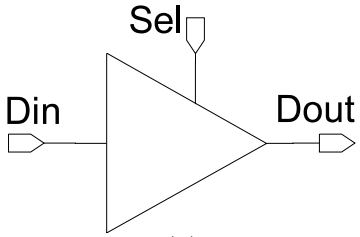
Neka datoteka `modelTop.vhdl` između ostalog sadrži i model ispitnog okruženja čiji entitet ima naziv `modelTop_t`. Testiranje modela se tada može obaviti simulacijom ispitne komponente, naredbom:

```
$ vhdle modelTop_t
```

Svako uspješno testiranje potrebno je pokazati asistentu.

1.2 Pogonski sklop s tri logička stanja

Oblikovati i testirati VHDL model sklopa čiji su simbol i tablica istine dani slikom 1.1. Rješenje zadatka (model sklopa i ispitno okruženje) je prikazano na sl.1.2.



(a)

Din	Sel	Dout
0	0	Z
0	1	0
1	0	Z
1	1	1

(b)

Slika 1.1: Simbol pogonskog sklopa sa tri stanja (a), i odgovarajuća tablica istine (b).

```
---------------------------
-- threeState1
---------------------------

library ieee;
use ieee.std_logic_1164.all;

entity threeState1 is
  generic(
    td_g: time :=5 ns
  );
  port(
    d_p:     std_logic;
    sel_p:   std_logic;
    z_p: out std_logic
  );
end entity threeState1;

architecture beh of threeState1 is
begin
  z_p <=
    d_p after td_g
      when (sel_p='1')
      else
    'Z' after td_g;
end architecture beh;



---------------------------
-- threeState1_t
---------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_textio.all;
use std.textio.all;

entity threeState1_t is
end entity threeState1_t;

architecture test of threeState1_t is
  signal d: std_logic;
  signal sel: std_logic;
  signal z: std_logic;

  procedure printStatus is
    variable l:line;
  begin
    write(l, now, right, 10, ns);
    write(l, string'("␣in:␣d="));
    write(l, d);
    write(l, string'(",␣sel="));
    write(l, sel);
    write(l, string'(",␣out:␣z="));
    write(l, z);
    writeline(output, l);
  end procedure printStatus;

begin
  t: entity work.threeState1
    generic map(
      td_g => 5 ns
    ) port map (
      d_p   => d,
      sel_p => sel,
      z_p   => z
    );

  process begin
    d <= '0';
    sel <='1';
    wait for 10 ns; printStatus;
    sel <='0';
    wait for 10 ns; printStatus;

    d <= '1';
    sel <='1';
    wait for 10 ns; printStatus;
    sel <='0';
    wait for 10 ns; printStatus;

    wait;
  end process;
end architecture test;
```

Slika 1.2: Model N-bitovnog pogonskog sklopa s tri stanja i odgovarajuće ispitno okruženje.

1.3 Potpuno zbrajalo

Napisati i testirati ponašajni VHDL model za jednobitno potpuno zbrajalo prema sljedećim uputama.

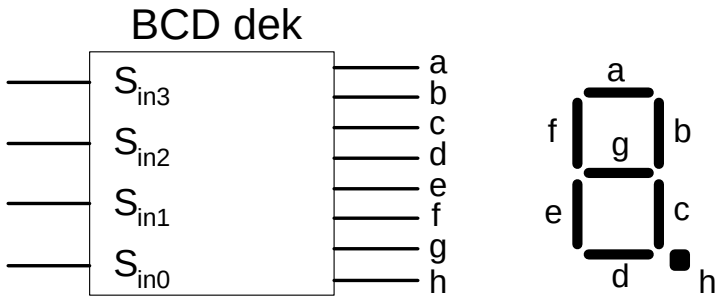
- napisati tablicu istine za kombinacijski sklop;
- oblikovati sučelje komponente (`entity`) koristeći tip `std_logic`;
- u izvedbi komponente (`architecture`), definirati stanje izlaznih linija za svaku kombinaciju stanja na ulazu korištenjem sljedne naredbe `case`;
- kašnjenje se može zanemariti;
- izlaz za ulazne vrijednosti koje nisu `'0'` ili `'1'` može biti nedefiniran (`'U'` ili `'X'`);
- modelirati ispitno okruženje kojim se ispituje funkcionalnost sklopa.

1.4 Multiplekser "4 na 1"

Koristeći upute zadatka 1.3, napisati i testirati ponašajni VHDL model za multiplekser "4 na 1".

1.5 BCD dekoder za 8-segmentni LCD sklopom

Koristeći upute iz zadatka 1.3, napisati i testirati ponašajni VHDL model za sklop koji upravlja 8-segmentnim prikaznim sklopom u ovisnosti o ulaznom BCD broju.



Slika 1.3: Sučelje dekodera za 8-segmentni prikazni sklop.

Struktura tablice istine dekodera je dana tablicom 1.1.

1.6 Bridom okidani bistabil

Oblikovati ponašajni VHDL model bridom okidanog JK bistabila. Provjeriti ispravnost modela ispitnim programom.

Upute:

ulazni broj	a	b	c	d	e	f	g	h
$0000_{(2)}$	1	1	1	1	1	1	0	0
$0001_{(2)}$								
$0010_{(2)}$	.							
$0011_{(2)}$	.							
$0100_{(2)}$	.							
$0101_{(2)}$								
$0110_{(2)}$								
$0111_{(2)}$								
$1000_{(2)}$								
$1001_{(2)}$								
ostali	0	0	0	0	0	0	0	1

Tablica 1.1: Skica tablice istine dekodera sa sl.1.3.

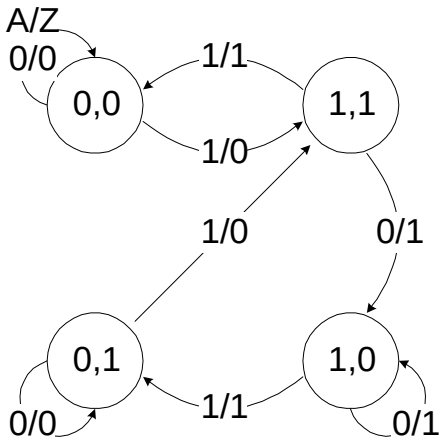
- sekvencijski sklopovi se modeliraju vrlo slično kombinatornim sklopovima, jer se mogu opisati kombinatornim sklopom koji na ulazu dobiva i ulazne vrijednosti i stanje sekvencijalnog sklopa;
- stanje sekvencijskog sklopa se opisuje unutrašnjim signalima komponente, koji se deklariraju u zaglavlju izvedbe (`architecture`) sklopa;
- kombinatorni sklop može opisivati samo promjene stanja jer se jednom postavljena vrijednost signala ne mijenja do sljedećeg pridruživanja.

Vježba 2

Ponašajni model sustava

2.1 Sekvencijski sklop

Modelirati i testirati sekvencijski sklop sa četiri stanja na temelju zadanog dijagrama (A je ulaz, a Z je izlaz). Sve promjene u sklopu se trebaju događati sinkrono s rastućim bridom signala vremenskog vođenja `clk`. Neka sučelje sklopa ima i inicijalizacijski ulaz `reset`, na čijem rastućem bridu se sklop prebacuje u početno stanje `0,0` uz vrijednost izlaza `Z=0`.



Slika 2.1: Dijagram stanja sekvencijskog sklopa.

2.2 Posmačni registar

Zadano je sučelje posmačnog registra koje sadrži dvije upravljačke linije `sel1` i `sel0` čije je značenje dokumentirano ispisom 2.1:

```
-- operation table
-- sel_p(1) sel_p(0)  operation
-- ==============================
--    0       0         nop
--    0       1         shift right
--    1       0         shift left
--    1       1         load
```

Ispis 2.1: Značenje upravljačkih linija posmačnog sklopa.

Pored upravljačkih linija, neka sučelje sadrži i signal vremenskog vođenja, serijski ulaz te paralelne ulazne i izlazne podatkovne linije. Definicija takvog sučelja je, uz parametrizaciju po broju bitova posmačnog sklopa, dana ispisom 2.2.

```
entity shifter is
  generic(
    cb_g: integer :=8;  -- count of bits
    td_g: time :=100 ns -- delay
  );
  port(
    sel_p: in  std_logic_vector(1 downto 0);
    sin_p: in  std_logic;                          -- serial input
    clk_p: in  std_logic;                          -- clock sync
    d_p:   in  std_logic_vector(cb_g-1 downto 0);  -- parallel input
    q_p:   out std_logic_vector(cb_g-1 downto 0)   -- parallel output
  );
end entity shifter;
```

Ispis 2.2: Sučelje posmačnog sklopa.

Oblikovati ponašajni model zadanog posmačnog sklopa u skladu sa sljedećim uputama:

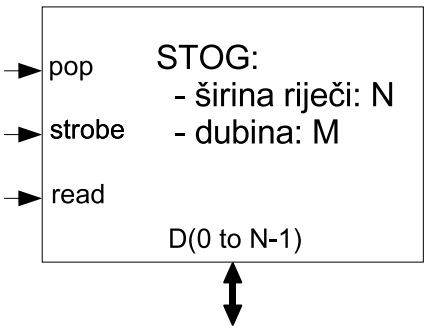
- deklarirati unutrašnji signal `reg_s` tipa `std_logic_vector`, u kojem će se čuvati podatak koji je trenutno upisan u registru;
- izvedbu temeljiti na procesu koji na temelju upravljačkih signala i stare vrijednosti unutrašnjeg signala `reg_s` određuje njegovu novu vrijednost;
- operatori posmaka nisu definirani za tip `std_logic_vector` pa je posmak potrebno modelirati konkatenacijom: npr. ako je vektor `vec` deklariran kao varijabla, njegov posmak za jedno mjesto ulijevo se postiže sa `vec := vec(vec'length-2 downto 0) & '0'`;
- konačno, proslijediti podatak iz registra od unutrašnjeg signala prema vanjskom korištenjem usporedne naredbe pridruživanja.

Ispravnost izrađenog modela je potrebno ispitati simulacijom, u kombinaciji sa odgovarajućim ispitnim okruženjem.

2.3 Sklopovski stog

Oblikovati ponašajni model sklopovske izvedbe stoga čije sučelje je zadano slikom 2.2.
Pri oblikovanju modela koristiti sljedeće upute:

- neka dubina stoga i širina riječi budu parametri sklopa;
- upis podatka na vrh stoga se postiže stavljanjem podatka na sabirnicu `D` te postavljanjem `pop='0'`, sinkrono s uzlaznim bridom signala `strobe`;
- skidanje podatka sa vrha stoga se postiže postavljanjem `pop='1'`, sinkrono s uzlaznim bridom signala `strobe`;



(a)

```
-- operation table
--   pop   strobe  operation
-- ==========================
--    0     0->1    push in (shift right)
--    1     0->1    pop out (shift left)

-- operation table
--   read   operation
-- ==========================
--    0      data <= Z
--    1      data <= top
```

(b)

Slika 2.2: Simbol sklopovskog stoga (a), i značenje pojedinih upravljačkih linija (b).

- čitanje podatka sa vrha stoga se postiže uz `read='1'`, kada sklop treba propustiti podatak s vrha stoga na sabirnicu `D`;
- ako ne teče operacija čitanja (`read='0'`), stog mora držati sabirnicu `D` u stanju visoke impedancije;
- izvedbu stoga temeljiti na dvodimenzionalnom polju:
  `type matrix is array(0 to N-1, 0 to M-1) of std_logic;`
  te operacijama posmaka ulijevo (pop) i udesno (push).

Ispravnost izrađenog modela je potrebno ispitati simulacijom, u kombinaciji sa odgovarajućim ispitnim okruženjem.

Vježba 3

Strukturno modeliranje

3.1 Naredba `generate`

Naredba `generate` omogućava opisivanje složenih sustava sa pravilnom strukturom. Predviđene su dvije sintakse naredbe, tzv. `for` i `if` oblik, kao što je prikazano ispisom 3.1.

```
stavak_generate ⇐
  labela:
  ( for parametar in <interval> | if <logički_izraz> )
    [ { <deklaracije> }
  begin]
    { <usporedne_naredbe> }
end generate [ labela ];
```

Ispis 3.1: Skica sintakse naredbe `generate`.

Često se koristi najjednostavniji slučaj, kad je jednu komponentu potrebno replicirati proizvoljno mnogo puta. Tada se koristi `for` oblik u kombinaciji sa instanciranjem komponente kao jedinom naredbom u tijelu naredbe `generate`. Kao primjer takvog korištenja naredbe generate, sl.3.1 prikazuje strukturni model pogonskog sklopa s tri logička stanja, s proizvoljnim brojem podatkovnih linija. Kao temeljnu građevnu jedinicu, model koristi jednobitni pogonski sklop opisan u vježbi 1.2. Obratiti pažnju na parametrizirano sučelje komponente, upotrebu naredbe `generate` te međusobnu sličnost ispitnih okruženja komponenti `threeState1` i `threeStateN`.

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

entity threeStateN is
  generic(
    td_g: time :=5 ns;
    -- count of bits
    cb_g: positive :=3
  ); port(
    d_p:     std_logic_vector
             (0 to cb_g-1);
    sel_p:   std_logic;
    z_p: out std_logic_vector
             (0 to cb_g-1)
  );
end entity threeStateN;

architecture struct of
  threeStateN
is
begin
  ts_vec: for i in 0 to cb_g-1
  generate
    one_ts:
      entity work.threeState1
      port map(
        d_p   => d_p(i),
        sel_p => sel_p,
        z_p   => z_p(i));
  end generate;
end architecture struct;
```

(a)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_textio.all;
use std.textio.all;

entity threeStateN_t is
end entity threeStateN_t;

architecture test of threeStateN_t is
  constant N: integer :=4;
  signal d: std_logic_vector(0 to N-1);
  signal sel: std_logic;
  signal z: std_logic_vector(0 to N-1);

  procedure printStatus is
    variable l:line;
  begin
    write(l, now, right, 10, ns);
    write(l, string'("␣in:␣d="));
    write(l, d);
    write(l, string'("␣in:␣sel="));
    write(l, sel);
    write(l, string'(",␣out:␣z="));
    write(l, z);
    writeline(output, l);
  end procedure printStatus;

begin
  t: entity work.threeStateN
    generic map(
      td_g => 5 ns,
      cb_g  => N)
    port map (
      d_p   => d,
      sel_p => sel,
      z_p   => z);

  process begin
    d <= "1010";
    sel <='1';
    wait for 10 ns; printStatus;

    sel <='0';
    wait for 10 ns; printStatus;

    d <= "1011";
    sel <='1';
    wait for 10 ns; printStatus;

    wait;
  end process;
end architecture test;
```

(b)

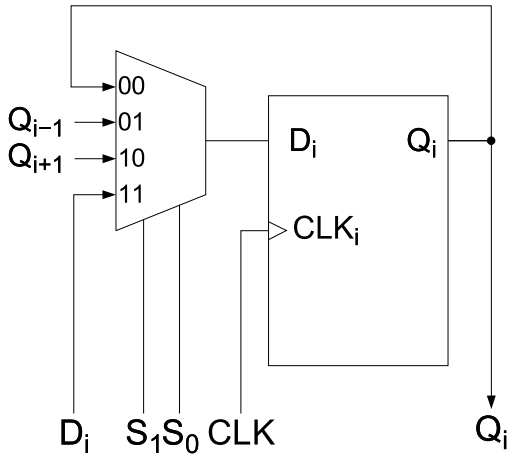
Slika 3.1: Model N-bitovnog pogonskog sklopa s tri stanja (a) i odgovarajuće ispitno okruženje (b).

3.2 Strukturni model zapornog sklopa

Oblikovati strukturni model zapornog sklopa uz parametrizaciju po broju bitova podatkovne riječi. Neka sklop ima paralelni podatkovni ulaz i izlaz, te upravljački signal na čijem se uzlaznom bridu podatak sa ulaza upisuje u sklop. Izvedbu temeljiti na naredbi `generate` i ponašajnom modelu D bistabila iz vježbe 1.6. Napisati odgovarajući ispitni program, te simulacijom provjeriti ispravnost modela.

3.3 Strukturni model posmačnog sklopa

Oblikovati strukturni opis posmačnog sklopa čije sučelje je zadano u vježbi 2.2. U izvedbi koristiti komponente mulitplekser "4 na 1" i bistabil čiji ponašajni modeli su razvijeni u okviru vježbi 1.4 odnosno 1.6, te naredbu `generate`. Shema spajanja komponenti za $i-$ti bit posmačnog sklopa je prikazana na sl.3.2.



Slika 3.2: Shema spajanja i-tog bistabila posmačnog sklopa.

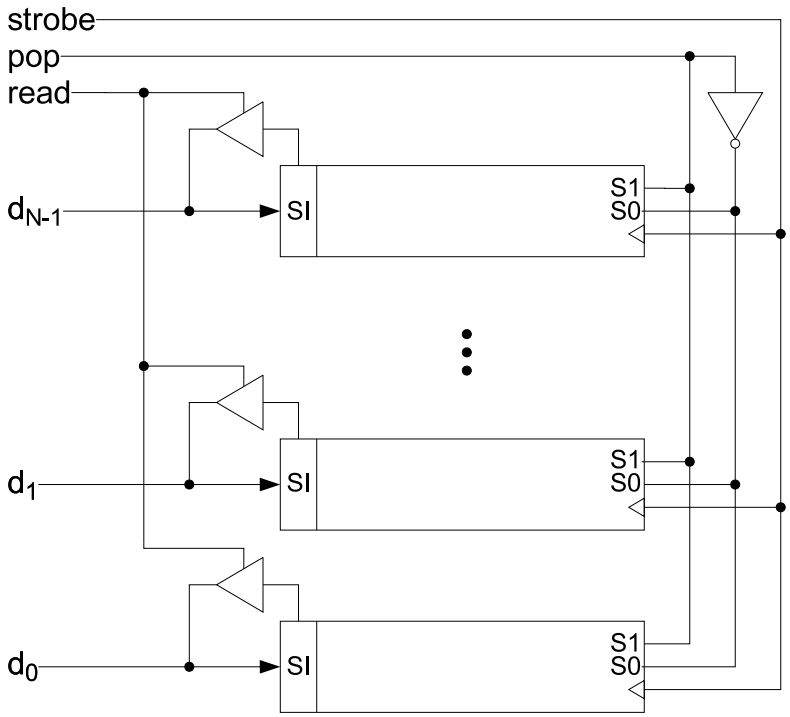
Pri oblikovanju slijediti upute:

- neka sučelje sklopa ostane kao i u vježbi 2.2, tako da se testiranje izvedbe može obaviti istim ispitnim programom;
- temeljni građevni blok sklopa (sl.3.2) opisati zasebnom komponentom;
- najlijeviji i najdesniji blok instancirati ručno kao specijalan slučaj, a sve ostale blokove (po potrebi) instancirati `for` oblikom naredbe `generate`.

Vježba 4

Strukturni model sklopovskog stoga

Oblikovati strukturni model sklopovskog stoga koji je prikazan na sl.2.2 (vježba 2.3). Sučelje sklopa treba biti parametrizirano dubinom stoga `M` i širinom pohranjenih riječi `N`, pa je u izvedbi ponašajnog modela prikladno koristiti naredbu `generate`. Kao građevne elemente sklopa mogu se koristiti komponente iz prethodnih zadataka: pogonski sklop s tri logička stanja i posmačni sklop, kao što je prikazano na sl.4.1. Modelirati odgovarajuće ispitno okruženje te simulacijom testirati ispravnost opisanog sklopa.



Slika 4.1: Strukturna shema sklopovskog stoga.

Dodatak A

Teme za referate

A.1 VHDL

Za svaku temu, potrebno je tekst popratiti s nekoliko primjera te formalnom sintaksom jezičnog konstrukta koji se opisuje. Primjeri moraju biti potpuni i sintaksno ispravni tako da se na njih može odmah primijeniti VHDL prevodioc. Referati se trebaju odnositi na standard iz 1993.

- Složeni tipovi polje (`array`) i zapis (`record`) te sintaksa za specificiranje složenih objekata (`aggregate`);
- Naredba assert, modeli kašnjenja.
- Predefinirani atributi tipova, polja i signala.
- Jednostavni predefinirani (`bit`, `bit_vector`, `boolean`, `character`, `int` i `real`), fizički i korisnički tipovi.
- Slijedne i usporedne uvjetne naredbe: `if`, `case`, `when`, `select`.
- Slijedne naredbe za ostvarivanje petlji `while`, `loop`, `for`, `exit`, `next`.
- Izravno `entity` i posredno `component`, `configuration` instanciranje komponenti.
- Temeljni operatori (aritmetički, logički, posmačni, povezivanje), konstantne vrijednosti (`literal`) i prazna naredba (`null`).
- Načini i primjena pristupa datotekama `file`.
- Funkcije `function` i procedure `procedure`.
- Korištenje procesa `process` te naredba `generate`.
- Parametri (`generic`) i konstante (`constant`).
- Pravila za pisanje i upotrebu bibliotečnih paketa (`package`).
- Paket std_logic_1164 te funkcija razrješavanja.
- Slijedne (*engl.* sequential) naredbe.

- Usporedne (*engl.* concurrent) naredbe.
- Korištenje aliasa (`alias`).
- Dijeljene varijable (`shared`) te paket `textio`;

Podaci o VHDL-u se mogu naći na sljedećim adresama:

- `http://tech-www.informatik.uni-hamburg.de/vhdl/`
- `http://www.angelfire.com/electronic/in/vlsi/vhdl.html`
- `http://www.eng.auburn.edu/department/ee/mgc/vhdl.html`
- `http://www.fh-pforzheim.de/fb05/mitarbeiter/becker/vhdl.pdf`