Projektiranje digitalnih sustava

vježbe iz VHDL-a

Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Fakultet elektrotehnike i računarstva

Sveučilište u Zagrebu

Primjer kombinatornog sklopa

Napisati i testirati VHDL ponašajni 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 usporedne naredbe **`select`**
- 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

Jednobitno zbrajalo - rješenje

```
----------------------------
-- adder_1
----------------------------

library ieee;
use ieee.std_logic_1164.all;

entity adder_1 is
  generic(
    td_g: time :=5 ns
  ); port(
    d2_p:     std_logic;
    d1_p:     std_logic;
    d0_p:     std_logic;
    s1_p: out std_logic;
    s0_p: out std_logic
  );
end entity adder_1;
```

Jednobitno zbrajalo - rješenje

```
-----------------------------
-- adder_1
-----------------------------

library ieee;
use ieee.std_logic_1164.all;

entity adder_1 is
  generic(
    td_g: time :=5 ns
  ); port(
    d2_p:     std_logic;
    d1_p:     std_logic;
    d0_p:     std_logic;
    s1_p: out std_logic;
    s0_p: out std_logic
  );
end entity adder_1;
```

```
architecture beh of adder_1 is
  signal input:  std_logic_vector(0 to 2
  signal output: std_logic_vector(0 to 1
begin
  input <= (d0_p,d1_p,d2_p);
  with input select
    output <=
      "00" after td_g when "000",
      "01" after td_g when "001",
      "01" after td_g when "010",
      "10" after td_g when "011",
      "01" after td_g when "100",
      "10" after td_g when "101",
      "10" after td_g when "110",
      "11" after td_g when "111",
      "UU" after td_g when others;
  (s1_p,s0_p) <= output;
end architecture beh;
```

Jednobitno zbrajalo - ispitna komponenta

```
----------------------------
-- adder_1_t
----------------------------

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

entity adder_1_t is
end entity adder_1_t;

architecture test of adder_1_t is
  -- deklaracije signala itd...
begin
  t: entity work.adder_1
    generic map(5 ns)
    port map (
      d0_p=>d0, d1_p=>d1,
      d2_p=>cin,
      s0_p=>s,  s1_p=>cout);
```

Jednobitno zbrajalo - ispitna komponenta

```
----------------------------
-- adder_1_t
----------------------------

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

entity adder_1_t is
end entity adder_1_t;

architecture test of adder_1_t is
  -- deklaracije signala itd...
begin
  t: entity work.adder_1
    generic map(5 ns)
    port map (
      d0_p=>d0, d1_p=>d1,
      d2_p=>cin,
      s0_p=>s,  s1_p=>cout);

  test: process
    constant base:integer :=
      std_logic'pos('0');
  begin
    for i in 0 to 1 loop
      for j in 0 to 1 loop
        for k in 0 to 1 loop
          d0  <= std_logic'val(base+i);
          d1  <= std_logic'val(base+j);
          cin <= std_logic'val(base+k);
          wait for 10 ns;
          printStatus;
        end loop;
      end loop;
    end loop;
    wait;
  end process;
end architecture test;
```

Prvi primjer sekvencijskog sklopa

Oblikovati VHDL ponašajni model za bridom okidani D bistabil te provjeriti ispravnost modela ispitnim programom.
Upute:

- 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.

Jednobitni b.o. bistabil - rješenje

```
----------------------------
-- ffD_1
----------------------------

library ieee;
use ieee.std_logic_1164.all;

entity ffD_1 is
  generic(
    td_g: time :=100 ns);
  port(
    d_p:   in  std_logic;
    clk_p: in  std_logic;
    q_p:   out std_logic);
begin
end entity ffD_1;
```

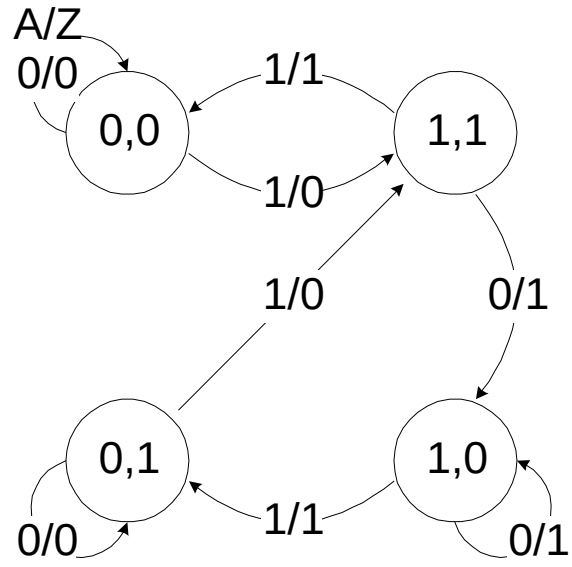
Jednobitni b.o. bistabil - rješenje

```
----------------------------
-- ffD_1
----------------------------

library ieee;
use ieee.std_logic_1164.all;

entity ffD_1 is
  generic(
    td_g: time :=100 ns);
  port(
    d_p:   in  std_logic;
    clk_p: in  std_logic;
    q_p:   out std_logic);
begin
end entity ffD_1;
```

```
architecture beh of ffD_1 is
begin
  ff: process(clk_p) begin
    if (rising_edge(clk_p)) then
      q_p <= d_p;
    end if;
  end process;
end architecture beh;
```

Drugi primjer sekvencijskog sklopa

- Modelirati i testirati sekvencijski sklop sa četiri stanja na temelju zadanog dijagrama
- 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`.

Automat s četiri stanja - rješenje (a)

```
-- library ...
entity dfa is
  generic(
    td_g: time :=100 ns
  ); port(
    reset_p: in  std_logic;
    a_p:     in  std_logic;
    clk_p:   in  std_logic;
    z_p:     out std_logic
  );
end entity dfa;

architecture beh of dfa is
  subtype state is
    std_logic_vector
    (1 downto 0);
  signal s_s: state;
```

Automat s četiri stanja - rješenje (a)

```
-- library ...
entity dfa is
  generic(
    td_g: time :=100 ns
  ); port(
    reset_p: in  std_logic;
    a_p:     in  std_logic;
    clk_p:   in  std_logic;
    z_p:     out std_logic
  );
end entity dfa;

architecture beh of dfa is
  subtype state is
    std_logic_vector
    (1 downto 0);
  signal s_s: state;
```

```
  -- procedura switchState
begin
  ff: process(clk_p, reset_p)
    variable snext: state;
    variable znext: std_logic;
  begin
    if (rising_edge(reset_p)) then
      z_p <= '0';
      s_s <= "00";
    elsif (rising_edge(clk_p)) then
      switchState(a_p, s_s, snext, znext
      z_p <= znext;
      s_s <= snext;
    end if;
  end process;
end architecture beh;
```

Automat s četiri stanja - rješenje (b)

```
procedure switchState(
  a: std_logic; s: state;
  snew: out state;
  znew: out std_logic)
is
  variable input:
    std_logic_vector
    (2 downto 0):=
    (s(1), s(0), a);
  variable output:
    std_logic_vector
    (2 downto 0);
```

Automat s četiri stanja - rješenje (b)

```
procedure switchState(
  a: std_logic; s: state;
  snew: out state;
  znew: out std_logic)
is
  variable input:
    std_logic_vector
    (2 downto 0):=
    (s(1), s(0), a);
  variable output:
    std_logic_vector
    (2 downto 0);
```

```
begin
  case input is
    when "000"  => output := "000";
    when "001"  => output := "110";
    when "010"  => output := "010";
    when "011"  => output := "110";
    when "100"  => output := "101";
    when "101"  => output := "011";
    when "110"  => output := "101";
    when "111"  => output := "001";
    when others => output := "UUU";
  end case;
  snew:=output(2 downto 1);
  znew:=output(0);
end switchState;
```

Izravno instanciranje komponenti

Modelirati D bistabil s ulazom za omogućavanje upotrebom prethodno modeliranog temeljnog D bistabila.

```
----------------------------
-- ffDe_1
----------------------------

library ieee;
use ieee.std_logic_1164.all;

entity ffDe_1 is
  port(
    d_p,clk_p,e_p: std_logic;
    q_p: out std_logic);
end entity ffDe_1;
```

Izravno instanciranje komponenti

Modelirati D bistabil s ulazom za omogućavanje upotrebom prethodno modeliranog temeljnog D bistabila.

```
----------------------------
-- ffDe_1
----------------------------

library ieee;
use ieee.std_logic_1164.all;

entity ffDe_1 is
  port(
    d_p,clk_p,e_p: std_logic;
    q_p: out std_logic);
end entity ffDe_1;
```

```
architecture struct of ffDe_1 is
  signal clke: std_logic;
begin
  clke <= clk_p and e_p;

  bb: entity work.ffD_1(beh)
    port map(
      d_p   => d_p,
      clk_p => clke,
      q_p   => q_p
    );
end architecture;
```

Naredba generate

- Naredba **generate** omogućava opisivanje složenih sustava sa pravilnom strukturom
- Predviđene su dvije sintakse naredbe, tzv. **for** i **if** oblik:

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

- 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**.

Primjer repliciranja jednostavnog sklopa

- Oblikovati strukturni model registra 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 temeljnog D bistabila
- Napisati odgovarajući ispitni program, te simulacijom provjeriti ispravnost modela.

Parametrizirani strukturni model registra

```
-------------------------
-- ffD_n
-------------------------

library IEEE;
use IEEE.Std_Logic_1164.all;

entity ffD_n is
  generic(
    cb_g: integer :=4;
    td_g: time :=100 ns
  ); port(
    d_p: in std_logic_vector
            (cb_g-1 downto 0);
    clk_p: in  std_logic;
    q_p:out std_logic_vector
            (cb_g-1 downto 0)
  );
end entity ffD_n;
```

Parametrizirani strukturni model registra

```
-------------------------
-- ffD_n
-------------------------

library IEEE;
use IEEE.Std_Logic_1164.all;

entity ffD_n is
  generic(
    cb_g: integer :=4;
    td_g: time :=100 ns
  ); port(
    d_p: in std_logic_vector
            (cb_g-1 downto 0);
    clk_p: in  std_logic;
    q_p:out std_logic_vector
            (cb_g-1 downto 0)
  );
end entity ffD_n;
```

```
architecture struct of ffD_n is
begin
  g_1 : for i in cb_g-1 downto 0
  generate
  begin
    onebit: entity work.ffD_1(beh)
      generic map(
        td_g => td_g
      ) port map(
        d_p   => d_p(i),
        clk_p => clk_p,
        q_p   => q_p(i)
      );
  end generate;
end architecture struct;
```

Složeniji parametrizirani strukturni model

- Oblikovati strukturni model sklopa za generiranje pariteta uz parametrizaciju po broju bitova podatkovne riječi
- Neka sklop ima sekvencijalnu strukturu sa n-1 sklopom "isključivo ili"
- Izvedbu temeljiti na naredbi **`generate`** i ugrađenom operatoru **`xor`**
- Napisati odgovarajući ispitni program, te simulacijom provjeriti ispravnost modela.

Parametrizirani model generator pariteta

```
--------------------------------
-- parity_n
--------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity parity_n is
    generic(
      n_g: integer :=10
    ); port(
      d_p: in std_logic_vector
              (0 to n_g-1);
      odd_p: out std_logic
    );
end parity_n;
```

Parametrizirani model generator pariteta

```
--------------------------------
-- parity_n
--------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity parity_n is
    generic(
      n_g: integer :=10
    ); port(
      d_p: in std_logic_vector
              (0 to n_g-1);
      odd_p: out std_logic
    );
end parity_n;
```

```
architecture seq of parity_n is
  signal p: std_logic_vector
            (0 to n_g - 3);
begin
  assert n_g>=3;

  p(0) <= d_p(0) xor d_p(1);

  G: for I in 1 to (n_g - 3) generate
    p(i) <= p(i-1) xor d_p(i+1);
  end generate G;

  odd_p <= p(n_g-3) xor d_p(n_g-1);
end seq;
```

Ispitna komponenta za generator pariteta

```
architecture test of parity_n_t is
  constant n: integer :=4;
  signal d: std_logic_vector
            (0 to n-1);
  signal odd: std_logic;

  type test_t is record
      d   : std_logic_vector
            (0 to n-1);
      odd : std_logic;
  end record;
  type test_vector_t is array
      (positive range <>) of test_t;
  constant testvec :
    test_vector_t := (
      ("0001",'1'),
      ("0101",'0'),
      ("0111",'1'),
      ("0011",'0')
    );
```

Ispitna komponenta za generator pariteta

```
architecture test of parity_n_t is
  constant n: integer :=4;
  signal d: std_logic_vector
            (0 to n-1);
  signal odd: std_logic;

  type test_t is record
      d   : std_logic_vector
            (0 to n-1);
      odd : std_logic;
  end record;
  type test_vector_t is array
      (positive range <>) of test_t;
  constant testvec :
    test_vector_t := (
      ("0001",'1'),
      ("0101",'0'),
      ("0111",'1'),
      ("0011",'0')
    );
```

```
begin
  uut: entity work.parity_n
    generic map(4)
    port map (d,odd);

  test: process
    variable test : test_t;
    variable l:line;
  begin
    for index in testvec'range loop
      test := testvec(index);
      d <= test.d;
      wait for 50 ns;

      -- printStatus
      if (test.odd /= odd) then
         assert false;
      end if;
    end loop;
    wait;
  end process;
end test;
```