Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

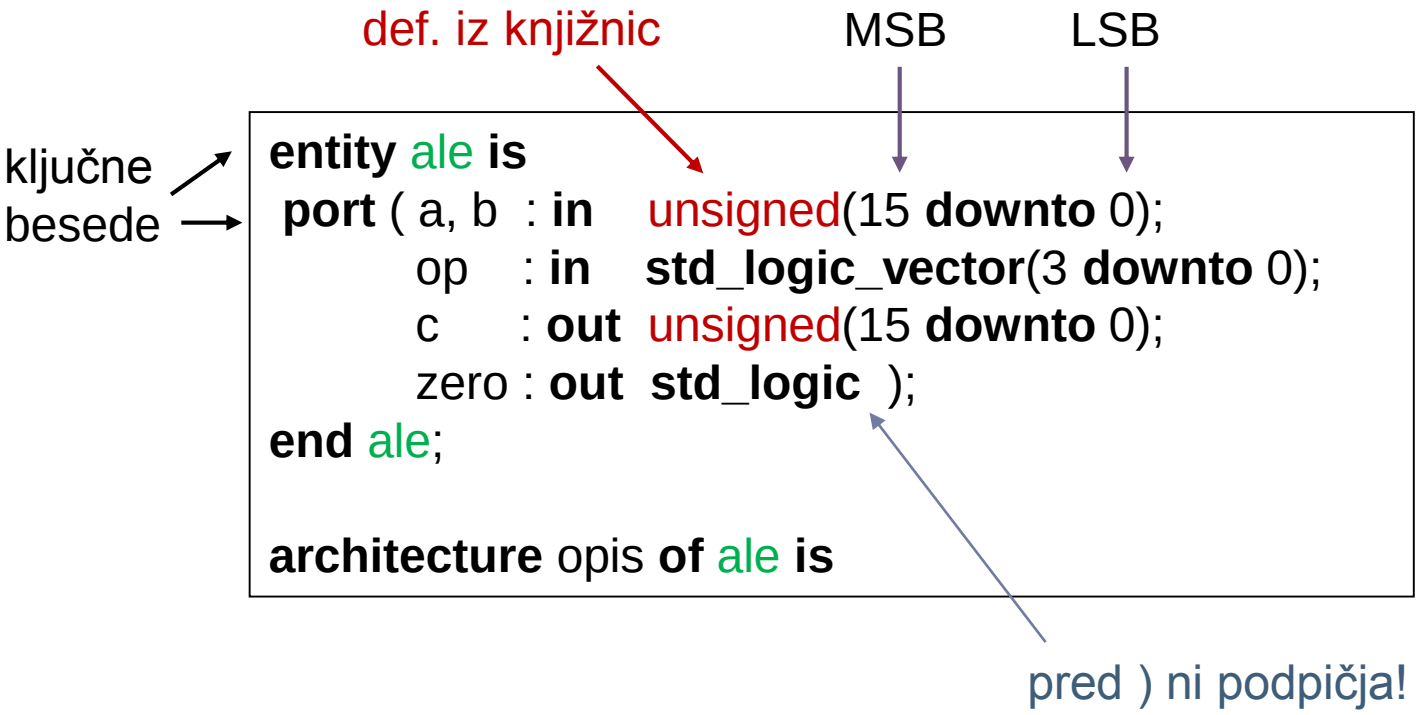
Digitalni Elektronski Sistemi

Osnove jezika VHDL

Povzetek

Entiteta (entity)

- Določi ime vezja, parametre in priključke
  - ime je identifikator iz znakov: a-z, A-Z, 0-9 in _
  - prvi znak je črka, zadnji ne sme biti podčrtaj, brez čšž in presledkov



Knjižnice

- na vrhu opisa vključimo IEEE standardne knjižnice za večvrednostno logiko in numerične operacije

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
```

- ne uporabljamo opuščenih (deprecated) knjižnic!

What are deprecated IEEE libraries?

Don't use `ieee.std_logic_unsigned` and similar libraries because they are not standardized. Instead, **use ieee.numeric_std.all**.

What's the difference between IEEE.STD_LOGIC_ARITH.ALL and IEEE.NUMERIC_STD.ALL?

std_logic_arith is a non-standard package created by Synopys back in the late '80s because at the time there was no standard way to do math on signals which were not declared as integer or natural. It was "put" into the IEEE library, even though it's not an IEEE standard.

That library has now been rightly deprecated in favor of **numeric_std**.

Notranji signali in konstante

- deklaracije signalov
  - začetne vrednosti določimo z :=

```
architecture one of test is

  signal count: unsigned(3 downto 0) := "0000";
  signal i :       integer := 0;
  signal data : signed(31 downto 0) := (others=>'0');

  constant zero: std_logic := '0';
  constant c1: signed(9 downto 0) := to_signed(150, 10);

begin
```

vse bite na '0'

pretvorba integer v signed

Operacije z logičnimi in numeričnimi signali

- logične operacije
  - **and, or, not, nand, nor, xor**

```
sum <= a xor b;
```

- vektorske operacije
  - **and, or, not, nand, nor, xor**
  - sestavljanje: a & b
  - podvektor: a(3 **downto** 0)
  - pomik: shift_right(a, 1)
  - razteg: resize(a, 10)

def. iz knjižnic za signed / unsigned

- aritmetične operacije
  - numerični signali
  - +, -, *

```
sum <= a + b;
inc <= a + 1;
m <= d * e;
```

podatkovni tip signed ali unsigned ujema se po velikosti (št. bitov)

- velikost rezultata je:

  +,- velikost večjega operanda

  * vsota velikosti operandov

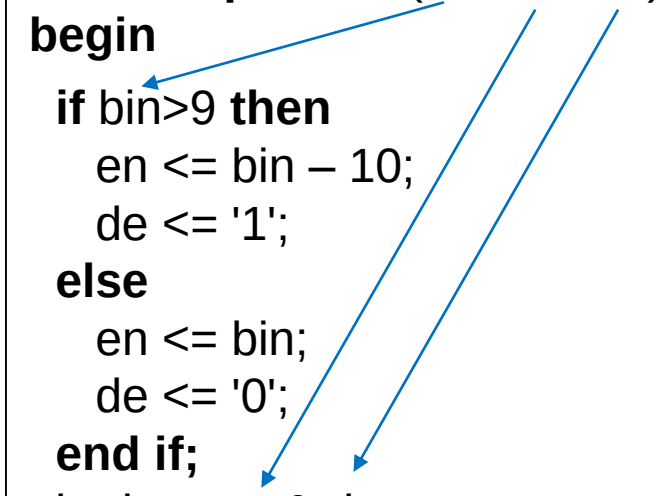
Kombinacijska logika

- pogojna prireditev

```
en <= bin - 30  when bin>20 else
      bin - 20  when bin>19 else
      bin - 10  when bin>9   else
      bin;
```

- kombinacijski proces
  - v oklepaju vsi signali, od katerih so odvisni izhodi procesa

```
bin2bcd: process(bin, en, de)
begin
  if bin>9 then
     en <= bin – 10;
     de <= '1';
  else
     en <= bin;
     de <= '0';
  end if;
  bcd <= en & de;
end process;
```



- izhod priredimo pri vseh komb. na vhodu (končni **else**)!

Sekvenčni gradniki

- proces s pogojem za fronto ure (rising_edge)
  - v oklepaju le signal ure


```
bcdstev: process (clk)
begin
 if rising_edge(clk) then
    if rst='1' then
       q <= (others=>'0');
    elsif q < 9 then
       q <= q + 1;
    else
       q <= "0000";
    end if;
 end if;
end process;
```

Opis vezja s komponentami

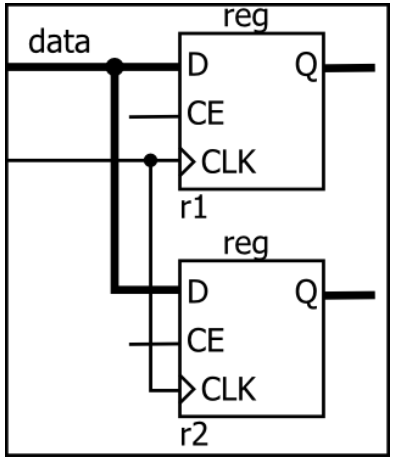
- deklaracija komponente in vključitev (povezava) komponent
  - component
  - port map (signal komp. => signal vezja)

```
architecture struktura of vezje is
  component reg is
    port ( d : in std_logic_vector(7 downto 0);
           clk, ce : in std_logic;
           q : out std_logic_vector(7 downto 0) );
  end component;

  signal en1, en2: std_logic;

begin
   r1: reg port map (d=>data, clk=>clk, ce=>en1, q=>data_reg);
   r2: reg port map (d=>data, clk=>clk, ce=>en2, q=>op_reg);
   ...

end struktura;
```



notranji signali

deklarirani priključki ali notranji signali vezja