

vaja 02

Proces

Digitalno načrtovanje – laboratorijske vaje
asistent: Nejc Ilc

Proces

```
<oznaka_procesa>: process (<sensitivity_list>)
begin
    -- stavki za opis procesa (zaporedni)
end
```

V `sensitivity_list` navedemo vse signale, ki lahko povzročijo spremembe na izhodih procesa.

Stavek "if"

- Obstaja samo znotraj procesa
- Sintaksa

```
if pogoj then
    -- stavki
else
    -- stavki
end if;
```

```
if pogoj_1 then
    -- stavki
elseif pogoj_2 then
    -- stavki
else
    -- stavki
end if;
```

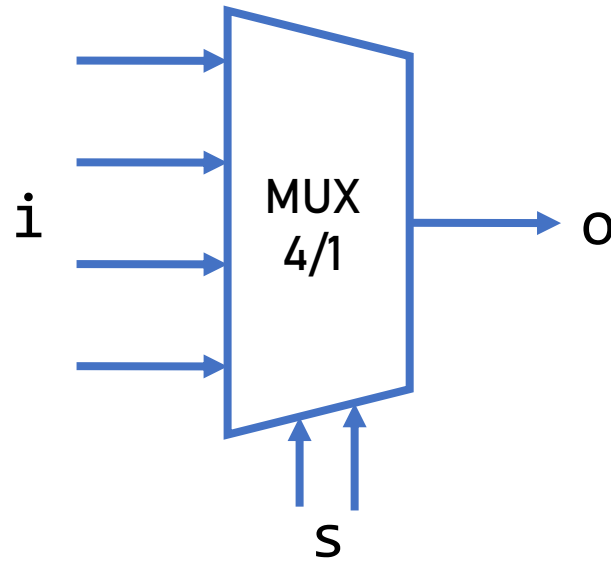
Stavek "case"

- Obstaja samo znotraj procesa
- Sintaksa

```
case s is
  when vred_1 => izhod <= izraz_1;
  when vred_2 => izhod <= izraz_2;
  ...
  when others => izhod <= izraz_dfl;
end case;
```

- Zadnji člen (`when others`) je obvezen.

Primer



S stavkom "case":

```
process (i, s)
begin
  case s is
    when "00" => o <= i(0);
    when "01" => o <= i(1);
    when "10" => o <= i(2);
    when "11" => o <= i(3);
    when others => o <= i(0);
  end case;
end process;
```

Ali s pogojnim prirejanjem:

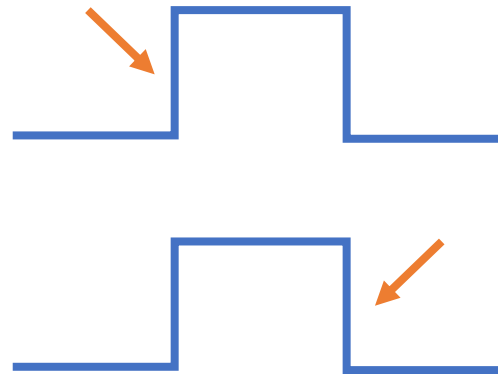
```
o <= i(0) when s="00" else
      i(1) when s="01" else
      i(2) when s="10" else
      i(3);
```

Ali s stavkom "select":

```
with s select
  o <= i(0) when "00",
      i(1) when "01",
      i(2) when "10",
      i(3);
```

Sekvenčna vezja

- V sekvenčnih vezjih se spremembe dogajajo ob dogodkih ure (ob prvi/pozitivni ali zadnji/negativni fronti)
- Dogodek na signalu ure `clk` zaznamo s `clk'event`
 - `clk'event` je `true`, če je prišlo do fronte
- Detekcija pozitivne fronte
`clk'event and clk='1'`
- Detekcija negativne fronte
`clk'event and clk='0'`



Primer: pomnilna celica D

```
process(clk)
begin
    if clk'event and clk = '1' then
        q <= d;
    end if;
end process;
```

Primer: števec

```
process(clk)
begin
    if clk'event and clk = '1' then
        if reset = '1' then
            count <= (others => '0'); -- reset all bits on '0'
        else
            count <= count + 1;
        end if;
    end if;
end process;
```


Knjižnica IEEE: paket STD_LOGIC_1164

- Knjižnica IEEE v paketu STD_LOGIC_1164 med drugim definira tipa STD_LOGIC in STD_LOGIC_VECTOR ter uporabni funkciji:
 - rising_edge
namesto clk'event and clk = '1'
 - falling_edge
namesto clk'event and clk = '0'

```
library IEEE;  
use IEEE.STD_LOGIC_1164.all;
```

Knjižnica IEEE: paket NUMERIC_STD

```
use IEEE.NUMERIC_STD.all;
```

- V paketu NUMERIC_STD sta med drugim definirani tipa `signed` in `unsigned` ter ustrezne aritmetične in logične operacije na njima:

`+`, `-`, `*`

`=`, `/=`, `<`, `<=`, `>`, `>=`

`shift_left(op1, op2)`, `shift_right(op1, op2)`

- pri tem je `op1` tipa `(un)signed` in `op2` tipa `integer`

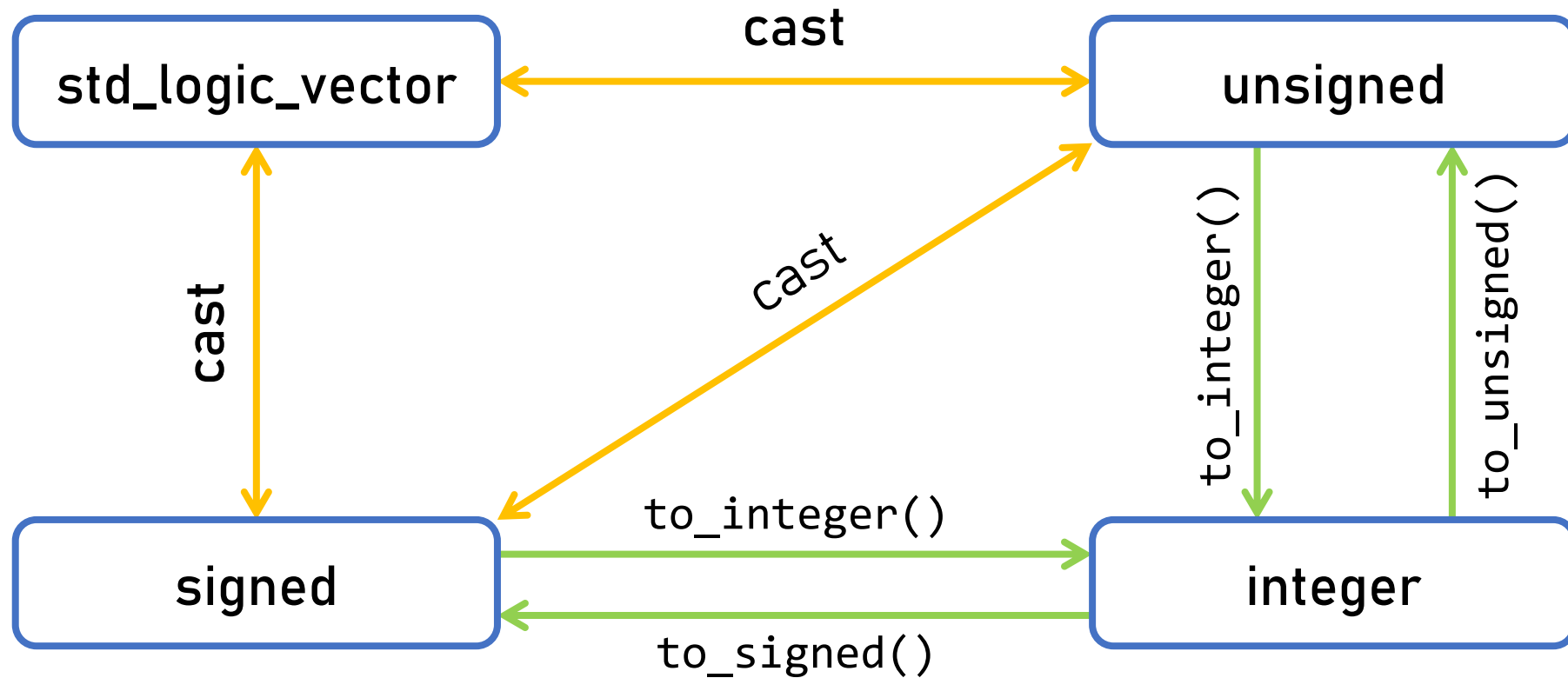
Knjižnica IEEE: paket NUMERIC_STD

Pretvorba podatkovnih tipov

- Pri uporabi aritmetičnih operatorjev se moramo pogosto posluževati pretvorbe tipov, npr.:
 - `std_logic_vector` \leftrightarrow `signed/unsigned`
 - `std_logic_vector` \leftrightarrow `integer`
 - `integer` \leftrightarrow `signed/unsigned`
- Uporabimo
 - reinterpretacijo (`cast`), npr.:
 - `unsigned(signal_std_logic_vector)`
 - `std_logic_vector(signal_unsigned)`
 - in funkcije za pretvorbo (`conversion`), npr.:
 - `to_integer(signal_unsigned)`
 - `to_signed(signal_integer, bit_width)`
- Primeri kode: <https://nandland.com/common-vhdl-conversions>

Knjižnica IEEE: paket NUMERIC_STD

Pretvarjanje podatkovnih tipov



Izziv

Napišite opis modula v VHDL, ki bo ustvaril vzorec „pomikanja“ na diodah LED:

- pomika se blok dolžine štiri (štiri prižgane LEDice)
- s prvim stikalom določamo smer pomikov
 - ko je stikalo vklopljeno, se smer pomikov obrne (leva \leftrightarrow desna)
- z drugim stikalom spreminjamo hitrost pomikov
 - ko je stikalo vklopljeno, se premiki dogajajo na 0,5 sekunde, sicer na 1 sekundo
- pritisk na sredinski gumb (BTNC) ponastavi vezje (reset)