



Laboratorij za načrtovanje integriranih vezij

Univerza v Ljubljani
Fakulteta za elektrotehniko



Digitalni Elektronski Sistemi

Osnove jezika VHDL

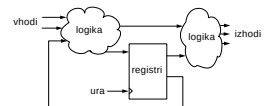
1. del: model vezja, signali in vektorji

I

Visokonivojski jeziki za opis vezij

- ▶ Standardizirani jeziki (IEEE standard)
 - ▶ VHDL (Very high-speed IC Hardware Description Language)
 - ▶ Verilog

- ▶ Modeliranje na različnih nivojih
 - ▶ logična vrata, RTL (Register Transfer Level)
 - ▶ nivo obnašanja (zapišemo algoritem)



- ▶ Funkcionalna in časovna simulacija vezja
 - ▶ s simulacijo preverimo delovanje vezja

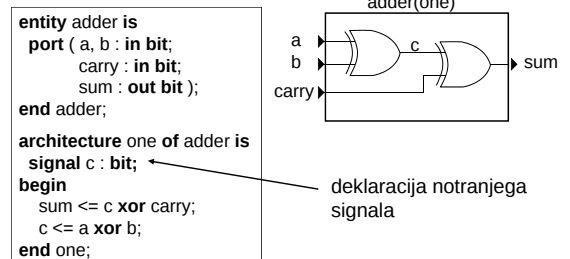
- ▶ Sinteza vezja
 - ▶ pretvorba vezja na nivo logičnih vrat in flip-flopov

Osnovni koncepti jezika VHDL

- ▶ Model vezja je sestavljen iz:
 - ▶ opisa vmesnika (entity)
 - ▶ opisa delovanja (architecture)
- ▶ Osnovni elementi opisa vezja so signali, ki predstavljajo povezave
- ▶ Pri opisu vmesnika določimo zunanje signale (port) in parametre vezja
 - ▶ zapišemo ime signala
 - ▶ določimo smer (in, out, inout, buffer)
 - ▶ in podatkovni tip (npr. bit)

```
entity adder is
  port ( a, b : in bit;
        carry : in bit;
        sum : out bit );
end adder;
```

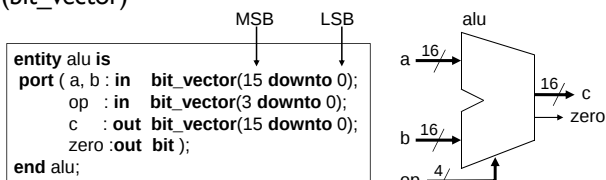
Opis delovanja (architecture)



- ▶ V arhitekturnem stavku določimo notranje in izhodne signale

Zunanji signali

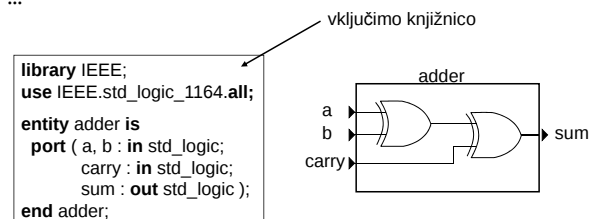
- ▶ Zunanji signali so enobitni (bit) ali vektorski signali (bit_vector)



- ▶ Podatkovni tip bit pozna dve vrednosti: 0 ali 1
 - ▶ za bolj realistično simulacijo potrebujemo več-vrednostno logiko ('Z', 'U', 'X')
 - ▶ potrebujemo tudi funkcije za razrešitev
 - ▶ npr. ko signalu vsilimo vrednost '0' in 'Z', prevlada vrednost '0'

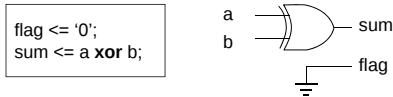
Večvrednostna logika: std_logic

- ▶ V knjižnici IEEE: std_logic in std_logic_vector
 - ▶ 'U' nedefinirano stanje
 - ▶ 'X' kratek stik
 - ▶ 'Z' visoka impedanca (odprte sponke)
 - ▶ 'H' pasivno visoko stanje (pullup)
 - ▶ ...



Prireditveni stavek

- ▶ S prireditvenim stavkom signalu priredimo konstantno vrednost ali izraz:

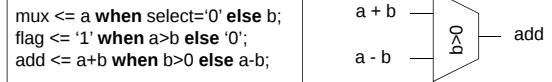


- ▶ Prireditveni stavek predstavlja logiko, katere izhod je signal, ki mu prirejamo vrednost

Za isti signal ne moremo imeti več sočasnih prireditvenih stavkov, ker bi opisali kratek stik!

Pogojni prireditveni stavek

- ▶ Signalu priredimo vrednost ali izraz ob določenem pogoju



- ▶ Pogojni prireditveni stavek v osnovni obliki predstavlja izbirnik (multiplexer)
 - ▶ izrazi predstavljajo logiko na vhodu izbiralnika, pogoj pa logiko za izbiro vhoda
- ▶ Relacijski operatorji:

=	/=	>	>=	<	<=
enako	ni enako	večje	večje ali enako	manjše	manjše ali enako

Konstantne vrednosti

- ▶ Konstantno vrednost tipa bit ali std_logic zapišemo med enojne narekovaje:

```
one <= '1';
tristate <= 'Z';
```

- ▶ Vektorske konstante (bit_vector ali std_logic_vector) imajo dvojne narekovaje:

```
a <= "11110000";
b <= X"3A";
c <= (others => '0');
```

šestnajstiška konstanta
agregat postavi vse bite vektorja na 0

Deklaracija signalov in konstant

- ▶ Zunanji signali so deklarirani v stavku port, notranji pa v arhitekturnem stavku
 - ▶ signalu lahko ob deklaraciji priredimo začetno vrednost
- ▶ Konstante deklariramo v arhitekturnem stavku

```
architecture one of test is
    signal flag: std_logic;
    signal reg: std_logic_vector(3 downto 0) := "0000";
    constant zero: std_logic := '0';
begin
```

Deklaracija zunanjih signalov

- ▶ Zunanji signali imajo poleg podatkovnega tipa določeno tudi smer

- in** ▶ vhodni signali, ki se lahko pojavljajo le na desni strani prireditvenih stavkov in v pogojih
- out** ▶ izhodni signali, ki se lahko pojavljajo le na levi strani prireditvenih stavkov
- inout** ▶ dvosmerni signali (npr. dvosmerna vodila), ki se lahko pojavljajo na obeh straneh prireditev
- buffer** ▶ izhodni signali, ki pa jih lahko uporabljamo tudi na desni strani prireditev in v pogojih

Primer deklaracije: buffer

```
entity increment is
    port ( a: in std_logic_vector(3 downto 0);
           incr_a : buffer std_logic_vector(3 downto 0);
           flag : out std_logic );
end increment;

architecture one of increment is
begin
    incr_a <= a + '1';
    flag <= '1' when incr_a="1111" else '0';
end one;
```

- ▶ Signal incr_a mora biti deklariran kot buffer, ker se pojavlja na levi strani prireditve (kot izhod) in v pogoju (kot vhod)

Podatkovni tipi: vektorji

- ▶ Vektorje uporabljamo za modeliranje večbitnih signalov (vodil)
- ▶ Območje indeksov običajno deklariramo od MSB proti LSB

```
signal a, b, c: std_logic_vector(7 downto 0);  
signal high, low: std_logic_vector(3 downto 0);
```

- Podvektor:

```
high <= a(7 downto 4); -- 4 bitni podvektor  
low <= a(3 downto 0); -- 4 bitni podvektor  
sign <= a(7); -- sign je tipa std_logic
```

Operacije z vektorji

- ▶ Logične operacije: **and, or, not, xor, xnor...**
- ▶ Sestavljanje vektorjev: **&**

```
a <= high & low;  
b <= sign & "000" & low;
```

- Pomikanje vektorjev
 - pomik naredimo s sestavljanjem in podvektorjem

```
b <= a(6 downto 0) & '0'; -- pomik v levo  
c <= '0' & a(7 downto 1); -- pomik v desno  
d <= a(6 downto 0) & a(7); -- rotacija v levo
```

Aritmetične operacije z vektorji

- Aritmetične operacije: **+, -, ***
 - vključimo knjižnjico: `std_logic_unsigned` (nepredznačena) ali `std_logic_signed` (predznačena števila)

```
library IEEE;  
use IEEE.std_logic_1164.all;  
use IEEE.std_logic_unsigned.all;  
  
architecture one of test is  
    signal a, b, sum, inc, m: std_logic_vector(7 downto 0);  
    signal d, e: std_logic_vector(3 downto 0);  
begin  
    sum <= a + b; -- 8 bitni seštevalnik  
    inc <= a + '1'; -- '1' se razširi na "00000001" in prišteje  
    m <= d * e; -- 4 x 4 biti = 8 bitni rezultat
```

Podatkovni tipi: integer

- ▶ Uporaba: indeksi vektorjev so cela števila
- ▶ Cela števila (integer) so predstavljena z 32 bitnim zapisom
- ▶ Pri deklaraciji lahko omejimo obseg vrednosti

```
architecture one of test is  
    signal a, b, c: integer;  
    signal d, e, f: integer range 0 to 255;  
begin  
    c <= a + b; -- 32 bitni seštevalnik  
    f <= d + e; -- 8 bitni seštevalnik
```

Izbira podatkovnih tipov

- ▶ Zunanji signali so vedno vektorji
- ▶ Notranji signali so lahko vektorji ali pa cela števila
- ▶ Za pretvorbo uporabimo ustrezne funkcije:

```
use IEEE.std_logic_arith.all; -- conv_std_logic_vector()  
use IEEE.std_logic_unsigned.all; -- conv_integer()  
  
architecture one of test is  
    signal i, j: integer;  
    signal a, b: std_logic_vector(7 downto 0);  
begin  
    i <= conv_integer(a); -- vektor v integer  
    b <= conv_std_logic_vector(j, 8); -- integer v vektor
```