VHDL jezik za opis hardvera

- Very high speed integrated circuits
- Hardware
- Description
- Language

VHDL jezik za opis hardvera

Standardizaciju je iniciralo i finansiralo Ministarstvo odbrane SAD-a u cilju efikasnijeg razvoja integrisanog kola.

VHDL jezik za opis hardvera

VHDL je 1987. godine usvojen kao standard:
IEEE Std. 1076-1987
VHDL-93,
VHDL AMS (Analogue and Mixed Signals) VHDL-99.

VHDL jezik za opis hardvera

Kasnije korekcije bile su ciljem da se ispoštuju zahtevi, odnosno primedbe korisnika i to 2000, 2002, 2007. i 2008, 2009. i 2011

VHDL jezik za opis hardvera

VHDL može da se primenjuje u svim fazama projektovanja:

- **opis,**
- **verifikaciju (simulaciju),**
- **sintezu i**
- **dokumentovanje**

VHDL semantika I sintaksa

- Deklaracija biblioteka sastoji se od liste svih biblioteka koje se koriste u projektu.
- Biblioteka se obično deklariše sa dve linije koda.
- Prva je ime biblioteke, a druga je njena upotreba koja počinje ključnom rečju **use**

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

VHDL semantika I sintaksa

- Definisanje logičkih stanja u okviru opisa svakog projekta samo bi nepotrebno opterećivalo kôd.
- Umesto toga, poziva se samo na biblioteka (*library*) koja sadrži te podatke.
- Cilj je da jezgro opisa bude maksimalno posvećeno konkretnom primeru, a da se sve ostalo nalazi u biblioteci.

VHDL semantika I sintaksa

- U okviru biblioteke, koju prepoznaje VHDL, nalazi se više celina koje se zovu paketi (**package**).
- Paketi, između ostalog, sadrže podatke o logičkim komponentama, familijama, serijama, konstantama, signalima, funkcijama I procedurama.

VHDL semantika I sintaksa

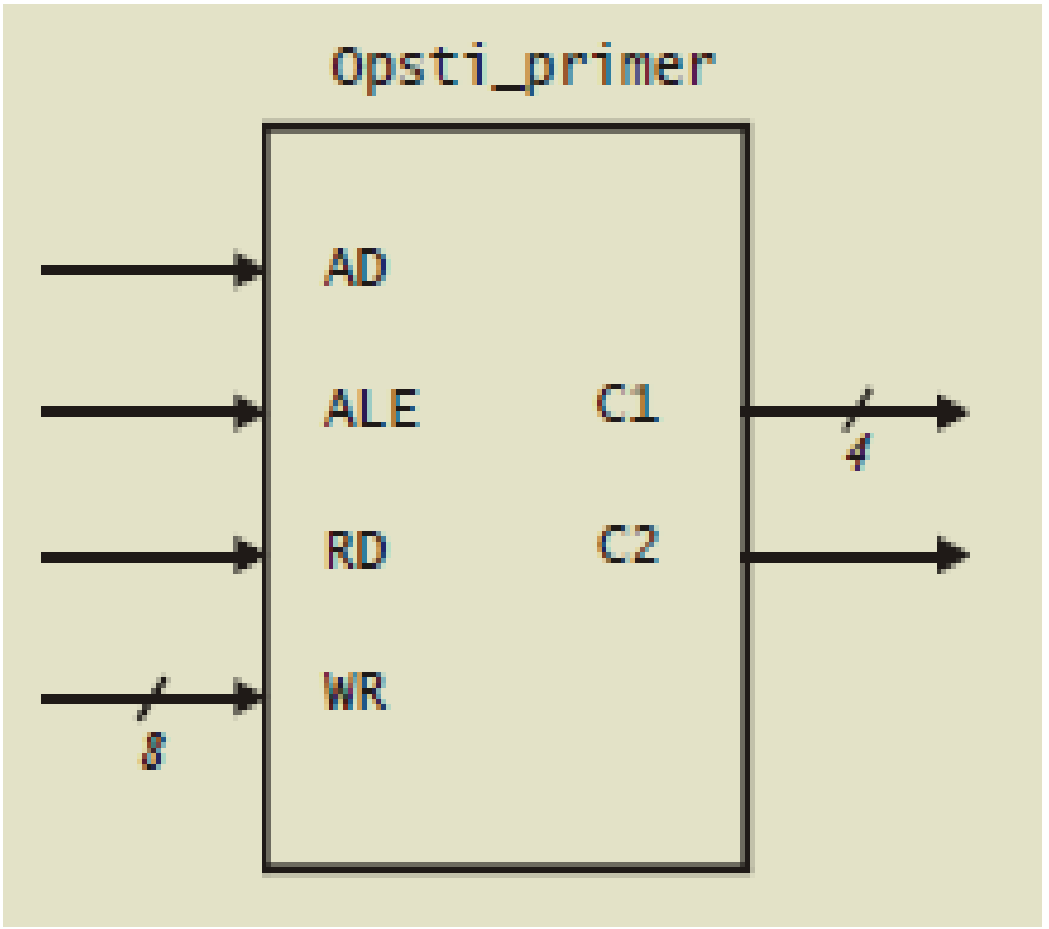
- Njima se pristupa pozivanjem iskaza use koji znači „iskoristi“.
- Sintaksa iskaza **use** zahteva da se navede naziv biblioteke, zatim paketa, I na kraju se navodi željeni pojam.
- Svi oni razdvojeni su tačkom:
  use <ime biblioteke>.<ime paketa>.<ime komponente>;

VHDL semantika I sintaksa

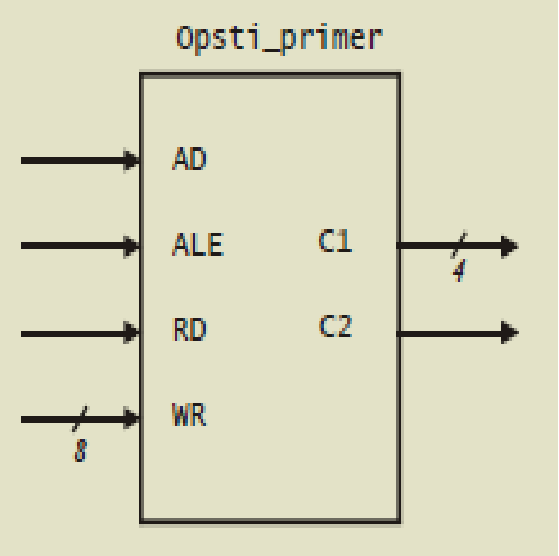
- Ukoliko umesto jednog pojma stoji ključna reč **all** (sve), to znači da se koriste svi pojmovi iz specificiranog paketa.
- Zapravo, kad god se koriste signali tipa *std_logic* čija se definicija nalazi u okviru standarda *IEEE Std. 1164*, koriste se biblioteka *IEEE* i paket *std_logic_1164* i to svi njeni elementi (**all**):

use IEEE. std_logic_1164.all;

VHDL Opis hardvera

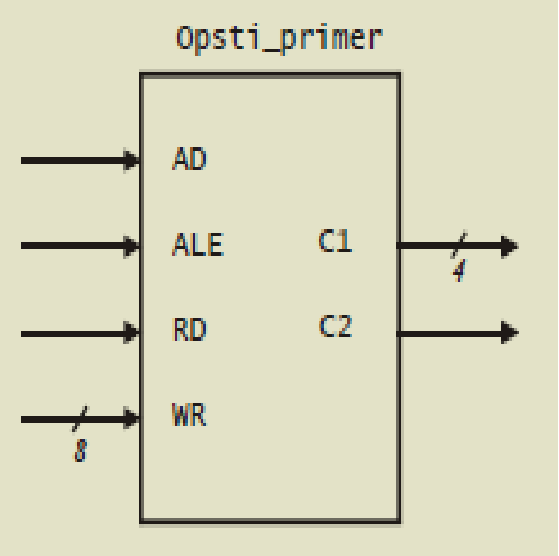


VHDL Opis hardvera



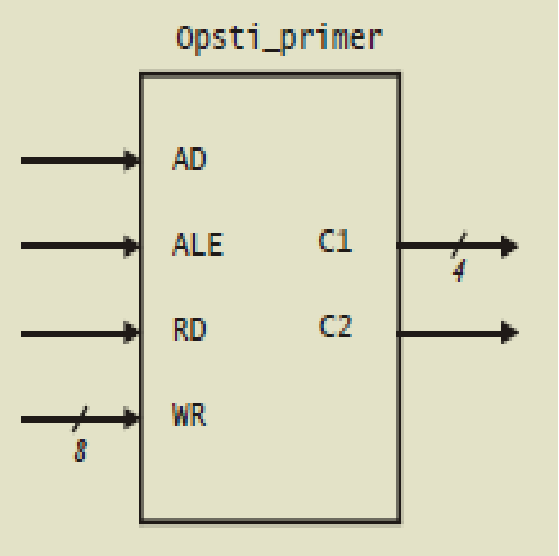
Svaki blok VHDL prepoznaje kao entitet koji se definiše preko ključne reči *entity*.

VHDL Opis hardvera



Da bi se entitetu dodelila funkcija koja povezuje stanja na ulaznim i izlaznim portovima potrebno je definisati arhitekturu entiteta. Opis arhitekture počinje ključnom reči *architecture*.

VHDL Opis hardvera

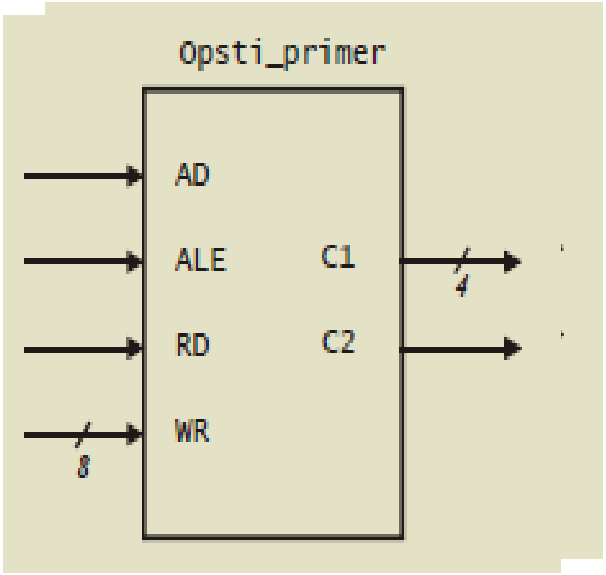



```
entity Opsti_primer is port(
AD,
ALE, RD ...
WR ...
C1 ...
C2 ...

);
end entity Opsti_primer
```

deklaracija portova

VHDL Opis hardvera




```
architecture proba of Opsti_primer
is
        .
        .
        .
begin
        .
        .
        .
end proba;
```

deklaracija signala

telo arhitekture

VHDL jezik za opis hardvera

- Kao što se u logickom projektovanju jedna ista funkcija može realizovati na više načina, tako i jednom entitetu može da se pridruži više arhitektura.
- Obrnuto ne važi jer jedna arhitektura može da obavlja samo funkciju zbog koje je projektovana.
- Očigledno je struktura VHDL jezika prilagodjena osnovnim strukturama koje se javljaju tokom projektovanja elektronskih kola.

VHDL signali

- Jedan entitet povezuje se sa drugim preko portova.
- Kroz portove putuju signali, a kako se radi o opisu digitalnih kola, reč je o digitalnim signalima.
- Zato i definicija signala u VHDL-u ima sve atribute koji se javljaju u realnim digitalnim kolima:
- *Vrednost*
- *Tip*
- *Mod*

VHDL signali

Svaki signal nosi *informaciju o logičkom stanju* ili logičkoj *vrednosti* **std_logic** *(*standard IEEE 1164):

vrednost	značenje
U	neinicirani signal (*Uninitialized*)
X	jako nepoznato stanje
0	jaka nula
1	jaka jedinica
Z	visoka impedansa
W	slabo nepoznato stanje (*Weak*)
L	slaba nula (*Low*)
H	slaba jedinica (*High*)
-	nebitno stanje (*don't care*)

VHDL signali

***Tip* signala:**

kroz jednu *žicu*, prolazi signal tipa bit *std_logic*,
kroz *magistralu*, signal je tipa vektor bitova *std_logic_vector*.

VHDL signali

Mod signala:

ulazni ***in***

Entitet može samo da čita sadržaj ovog signala, a ne može da mu dodeljuje novu vrednost

izlazni ***out***
Vrednost mu se dodeljuje unutar entiteta, a njegov sadržaj ne može da se koristi za pobudu drugih ulaza istog entiteta

bidirekcioni ***inout***

VHDL signali

Redosled bitova u signalu tipa std_logic_vector:

MSB levo:

std_logic_vector(3 downto 0)

0101 vrednost 5

MSB desno:

std_logic_vector (0 to 3)

0101 vrednost 10

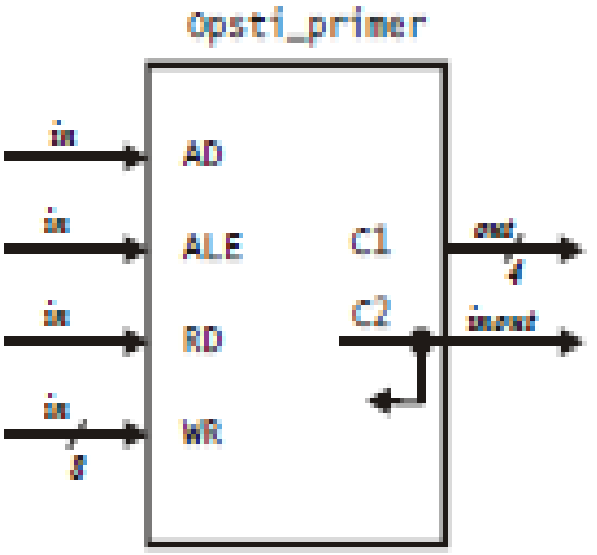
VHDL signali

Redosled navođenja :
naziv porta : mod tip

```
entity Opsti_primer is port(
AD:in std_logic;
ALE,RD:in std_logic;
WR    :in std_logic_vector(7 downto 0);
C1    :out std_logic_vector(3 downto 0);
C2    :inout std_logic_vector;
);
end entity Opsti_primer
```

VHDL signali

Redosled navođenja :
naziv porta : mod tip




```
entity Opsti_primer is port(
AD:in std_logic;
ALE,RD:in std_logic;
WR    :in std_logic_vector(7 downto 0);
C1    :out std_logic_vector(3 downto 0);
C2    :inout std_logic_vector;
);
end entity Opsti_primer
```

VHDL sintaksa

- **Svaki jezik karakteriše skup sintaksnih pravila za pisanje koda**
- **Ta pravila mogu da se podele u dve osnovne kategorije:**

-način označavanja pojmova
-format navodjenja pojmova.

VHDL sintaksa

Način označavanja pojmova:

- **Ne pravi se razlika izmedju malih i velikih slova**
- **Svi nazivi moraju da počnu slovnim znakom**
- **Koriste se samo slova (a-z i A-Z), brojevi (0-9) I podvučena crta (_)**
- **NE koriste se znakovi interpunkcije !, ?, ., ,, i.t.d.**
- **NE koristi se sukcesivno dva simbola (_)**

VHDL sintaksa

Ispravne oznake:

Prvi_primer
Opsti_Primer0
OPSTI_PRIMER_9
Opsti_Primer_0

Neispravne oznake:	**Razlog**
`1_Primer`	**pocinje brojem**
`Opsti_#1_primer`	**koristi se #**
`Opsti!_primer`	**koristi se !**
`Opsti__primer`	**koriste se dve _ _ sukcesivno**

VHDL sintaksa

Način označavanja pojmova (nastavak)

- **NE mogu dva različita pojma u okviru istog entiteta ili arhitekture imati ista imena, odnosno oznake**
- **Postoje rezervisane reči koje se NE SMEJU koristiti za označavanje pojmova (date u svakom priručniku za VHDL)**

VHDL kljucne reci

abs	downto	library	procedure	subtype
access	else	linkage	process	then
after	elsif	literal	pure	to
alias	end	loop	range	transport
all	entity	map	record	type
and	exit	mod	register	units
architecture	file	nand	reject	unaffected
array	for	new	rem	until
assert	function	next	report	use
attribute	generate	nor	return	variable
begin	generic	not	rol	wait
block	group	null	ror	when
body	guarded	of	select	while
buffer	if	on	severity	with
bus	impure	open	shared	xor
case	in	others	signal	xnor
component	inertial	out	sla	
configuration	inout	package	sll	
constant	is	port	sra	
disconnect	label	postponed	srl	

VHDL sintaksa

Način označavanja pojmova (nastavak)

Linijski komentari počinju dvostrukim znakom minus „--“.

```
entity Digitalni_modul is port

 ( ULAZ1: in std_logic; -- svaki port moze da se
            -- deklarise u posebnoj
                        -- liniji
    ULAZ2, ULAZ3: in std_logic; -- portovi istog
tipa
          -- mogu da se deklarisu u istoj liniji
-- tip bit-vektor mora da sadrzi
-- duzinu i redosled:
   ULAZ4: in std_logic_vector (7 downto 0);
-- 7 je MSB
   IZLAZ1: out std_logic_vector (0 to 3);
-- 0 je LSB
   IZLAZ2: inout std_logic    -- ovde nema ';'
       );
-- jer se ';' stavlja iza
-- zaokruzene logicke celine
-- koja sadrzi i zatvorenu zagradu
end entity Digitalni_modul;
```

VHDL sintaksa

Način označavanja pojmova (nastavak)

- **Ne postoji ograničenje u broju karaktera kojim se može označiti neki**
- **pojam.**
- **Međutim neki programi za sintezu prepoznaju najviše 32 karaktera.**
- **Zato se preporučuje da broj karaktera u jednoj oznaci ne prelazi 32.**
- ***Dobro je da* oznake budu dovoljno duge da ukažu na pravo značenje, ali da ne budu l *preduge.***
- ***Dobra je praksa da se signali nazovu* clock, data ili global_input, *a ne* c, d, ili g.**

VHDL sintaksa

Format pisanja VHDL-a treba sagledati sa stanovišta opisa pojedinih celina.

Pri tome celinu čini:

- **jedna naredba,**
- **struktura naredbi**
- **ili grupa naredbi**

VHDL sintaksa

Pod jednom *naredbom* podrazumevamo:
• opis koji počinje nekom od rezervisanih reči ili
• opis aktivnosti koja označava dodeljivanje vrednosti nekom signalu.

VHDL sintaksa

strukturu naredbi **čini više povezanih naredbi sa jasnim semantičkim značenjem na primer:**

if...then...elsif...else

```
if uslov1 then akcija1;
elsif uslov2 then akcija2;
else akcija3;
end if;
```

Svaka logička celina završava se simbolom „ **;** “.

VHDL sintaksa

Grupa naredbi obično počinje ključnom reči ili identifikatorom, a može da sadrži:

- skup naredbi kojima se deklarišu promenljive koje se javljaju u toj grupi naredbi,
- ključnu reč **begin** za označavanje početka opisa tela grupe naredbi,
- telo grupe naredbi I
- ključnu reč **end** kojom se opis grupe naredbi završava.

VHDL sintaksa

Grupom naredbi definišu se entiteti, arhitekture, procesi i neke druge celine.

• Iza **begin** očekuje se nastavak opisa, što znači da se iza ove reči ne očekuje znak „**;**“ koji označava završetak logičke celine.

• U okviru jedne logičke celine VHDL koristi „slobodni format“ za pisanje kôda,

• Ignorišu se prekidi linije i prazni karakteri.

VHDL sintaksa

```
if uslov1 then akcija1;     -- prva varijanta

if    uslov1  then   akcija1; -- druga varijanta

if uslov1
        then akcija1;     -- treca varijanta


if
uslov1
then akcija1;         -- cetvrta varijanta


if
uslov1
then              -- peta varijanta
```

VHDL osnovne logicke operacije

VHDL podržava korišćenje osnovnih logičkih operatora nad signalima tipa *std_logic*.

Operator	Klasa operatora
`**, abs, not`	Mešoviti operatori
`*, /, mod, rem`	Operatori množenja/deljenja
`+, -`	Operatori predznaka
`+, -, &`	Operatori zbrajanja
`=, /=, <, <=, >, >=`	Relacioni operatori
`SLL, SLR, SLA, SRA, ROL, ROR`	Operatori pomeranja
`AND, NAND, OR, NOR, XOR, XNOR`	Logički operatori

VHDL operatori

Dodeljivanje logičke vrednosti nekom signalu ili portu označava se simbolom

<=

Rezultat operacije nad većim brojem signala, npr. a i b može da se iskaže na sledeći način:

```
rezultat  <=  a AND b;
```

VHDL operatori

Od svih logičkih operatora NOT ima najveći prioritet, dok su svi ostali logički operatori istog prioriteta.
Redosled njihovog izvršavanja određuje se redosledom navođenja u naredbi.
Tako u naredbi:
rezultat **<=** a **AND** b **OR** c **XOR** d**;**

izvrši se najpre AND operacija, zatim OR i na kraju XOR, što može da se iskaže kao:

(((a AND b) OR c) XOR d).

VHDL operatori

Međutim u opisu:

rezultat <= a AND NOT b OR c XOR d;

redosled izvršavanja operacija je sledeci:

NOT, AND, OR, XOR:

(((a AND (NOT b)) OR c) XOR d)

VHDL redosled izvršavanja naredbi

- VHDL je namenjen za opis ponašanja u digitalnim sistemima.
- S obzirom da se aktivnosti u hardveru dešavaju uglavnom paralelno, konkurentno, i u opisu arhitekture nekog entiteta primenjuje se ista logika.
- Arhitektura može da sadrži više logičkih celina.
- Događaji iz jedne celine najčešće ne utiču na događaje u ostalim, odnosno odvijaju se nezavisno, konkurentno, tj. paralelno u vremenu. Međutim, postoje slučajevi kada su događaji međusobno uslovljeni.
- Tada jedan događaj pokreće naredni, odnosno postoji sekvenca
- događaja koju hardver izvršava u striktno definisanom redosledu.

VHDL redosled izvršavanja naredbi

Zato VHDL podržava dva osnovna metoda kroz koje se signalima unutar arhitekture dodeljuju vrednosti.
To su:
• konkurentna i
• sekvencijalna dodela vrednosti signalima.

VHDL redosled izvršavanja naredbi

- Operator dodeljivanja **<=** još se naziva naredba jednostavnog dodeljivanja (engl. *simple signal assignment*), jer se odnosi na dodeljivanje pojedinom signalu ili pojedinoj grupi signala.
- Osim ovog, VHDL podržava izborne (selekcione) i uslovne naredbe dodeljivanja.
- To su naredbe istovremenog ili konkuretnog dodeljivanja (engl. *concurrent assignment*).
- Njima se modeluju istovremene (paralelne) aktivnosti hardvera.

VHDL redosled izvršavanja naredbi

- Događaji iz jedne celine ne utiču na događaje iz ostalih celina.
- Drugim rečima, događaji se odvijaju nezavisno, tj. paralelno u vremenu. Zato i redosled njihovog navođenja, pri opisu arhitekture, nije važan.
- Ove naredbe, koriste ključne reči

with i **when**.

VHDL redosled izvršavanja naredbi

```
architecture proba of
 Opsti_primer is begin

  c2 <= ad OR ale;

  c1(0) <= rd AND wr(0);

  seq: process (wr(7))
   begin.
.
.
  end process seq;

end architecture proba;
```

=

```
architecture proba of
 Opsti_primer is begin

  c2 <= ad OR ale;

  seq: process (wr(7))
   begin.
.
.
  end process seq;

  c1(0)   <=   rd   AND
wr(0);

end          architecture
```

VHDL redosled izvršavanja naredbi

Naredbom izbornog dodeljivanja, signalu se dodeljuje vrednost uslova, a uslovi nakon ključne reči **when** moraju da budu međusobno Isključivi.

Sintaksa ove naredbe je sledeća:

with izraz **select**
    ime signala <= izraz **when** konstanta_vrednost {, izraz **when** konstanta_vrednost};

VHDL redosled izvršavanja naredbi

Semantika uslovnog dodeljivanja (engl. *conditional signal assignment*) proizilazi iz sekvencijalnog (slednog) karaktera izračunavanja, jer se signalu dodeljuje vrednost utvrđena izrazom uz istinit logički izraz uslova. Sintaksa ove naredbe je:

ime_signala <= izraz **when** logički_izraz **else** {, izraz **when** logički_izraz **else**} izraz;

VHDL redosled izvršavanja naredbi

Da bi se omogućio opis sekvencijalnih događaja, koji suštinski definišu određeni proces, uvedena je, kao posebna kategorija, grupa naredbi nazvana

process.

```
architecture proba of
    Digitalni_modul is
 begin
  IZLAZ2 <= ULAZ1 OR ULAZ2;
  IZLAZ1(0) <= ULAZ3 AND
            ULAZ4(0);
  seq: process (ULAZ4(7))
    begin  .
      .
      .
    end process seq;
end architecture proba;
```

=

```
architecture proba of
    Digitalni_modul is
 begin
  IZLAZ2 <= ULAZ1 OR ULAZ2;
  seq: process (ULAZ4(7))
    begin  .
      .
      .
    end process seq;
  IZLAZ1(0) <= ULAZ3 AND
            ULAZ4(0);
   end architecture proba;
```

VHDL redosled izvršavanja naredbi

Za razliku od konkurentnog dodeljivanja vrednosti signalima unutar arhitekture, dodeljivanje vrednosti unutar procesa obavlja se sekvencijalno po redosledu po kome su naredbe navedene.
Zato je redosled navođenja naredbi
unutar procesa kada se koriste promenljive, veoma bitan.

```
architecture proba of
       Digitalni_modul is
 begin
  seq: process
   (ULAZ1, ULAZ2, ULAZ3,
    ULAZ4)
    begin
     IZLAZ2 <= ULAZ1 OR
               ULAZ2;
     IZLAZ2 <= ULAZ3 AND
               ULAZ4(0);
    end process;
 end proba;
```

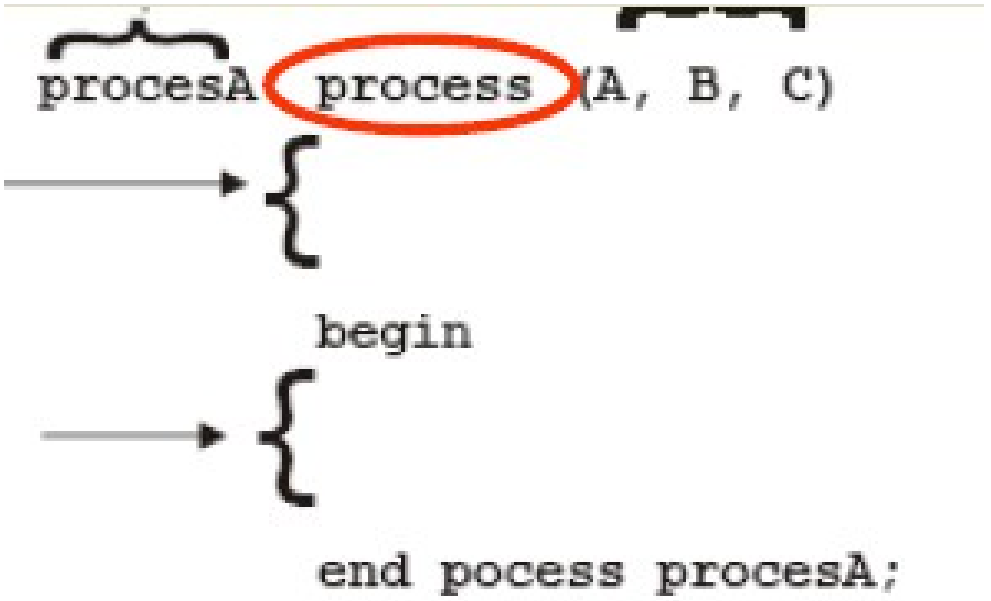
≠

```
architecture proba of
       Digitalni_modul is
 begin
  seq: process
   (ULAZ1, ULAZ2, ULAZ3,
    ULAZ4)
    begin
     IZLAZ2 <= ULAZ3 AND
               ULAZ4(0);
     IZLAZ2 <= ULAZ1 OR
               ULAZ2;
    end process;
 end proba;
```

VHDL proces

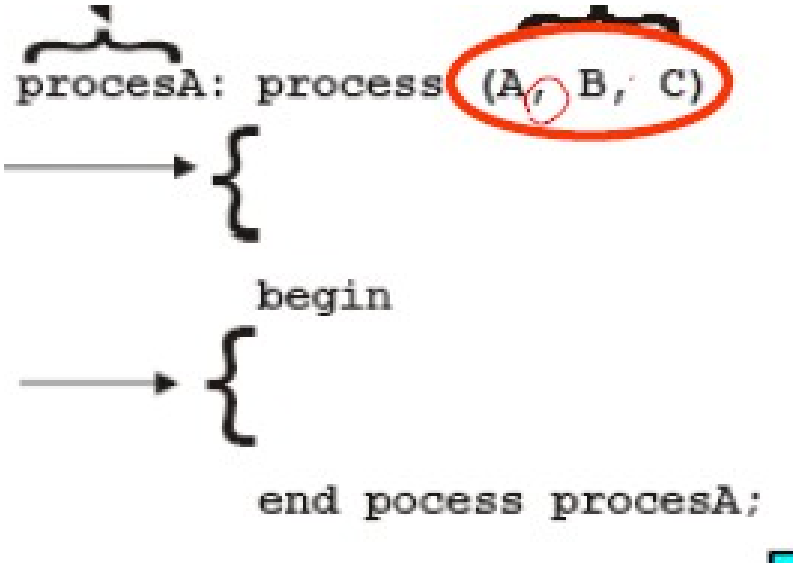
- Svaki proces treba da nosi jedinstvenu oznaku – labelu koja se završava sa “:”.
- Ona je opciona, ali se preporučuje naročito sa stanovišta sinteze.
- Signali koji se generišu automatskom sintezom sadržaće i ime procesa iz koga su proistekli čime se olakšava tumačenje celog projekta, naročito u fazi otkrivanja potencijalnih grešaka.
- Iza oznake (labele) procesa sledi ključna reč

process



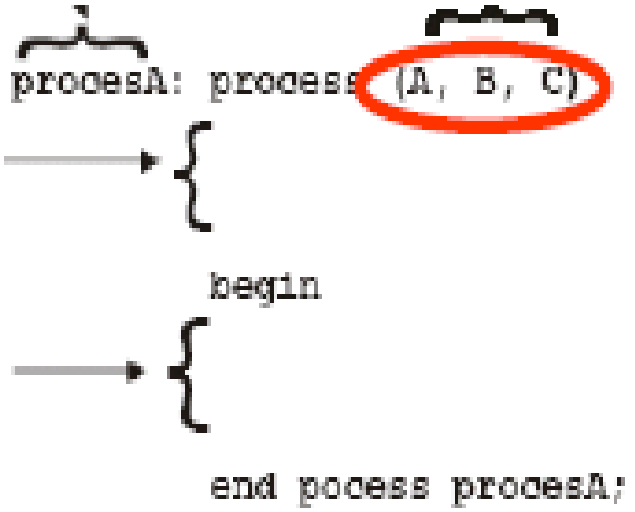
VHDL proces

- Zatim se navodi lista signala na koje je proces osetljiv.
- Lista signala navodi se između malih zagrada, avoznake pojedinih signala odvojene su zapetama.



VHDL proces

Ovaj oblik podrazumeva da jedan proces može da počne samo kada nastane promena u nekom od signala koji se u vidu liste navode na početku deklaracije proces.

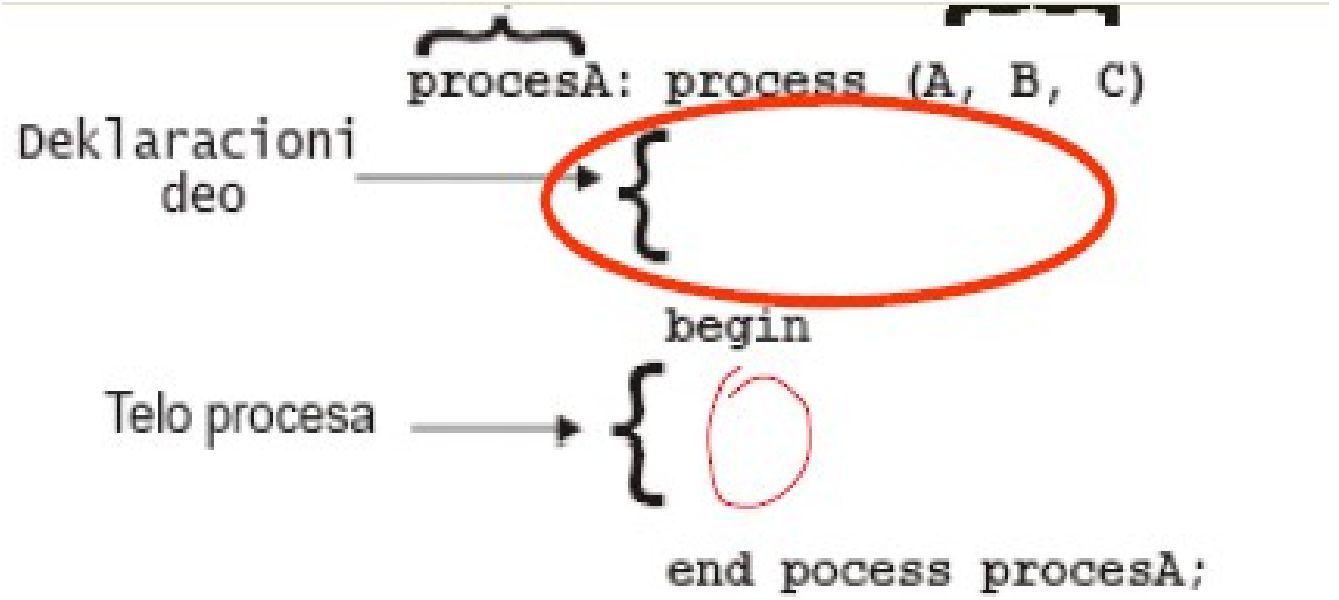


VHDL proces

Ovakav opis ekvivalentan je slučaju da se kao poslednje naredbe u procesu nalaze one koje **nalažu čekanje sve dok ne dođe do promene stanja bilo kog od signala na koje je proces osetljiv.**

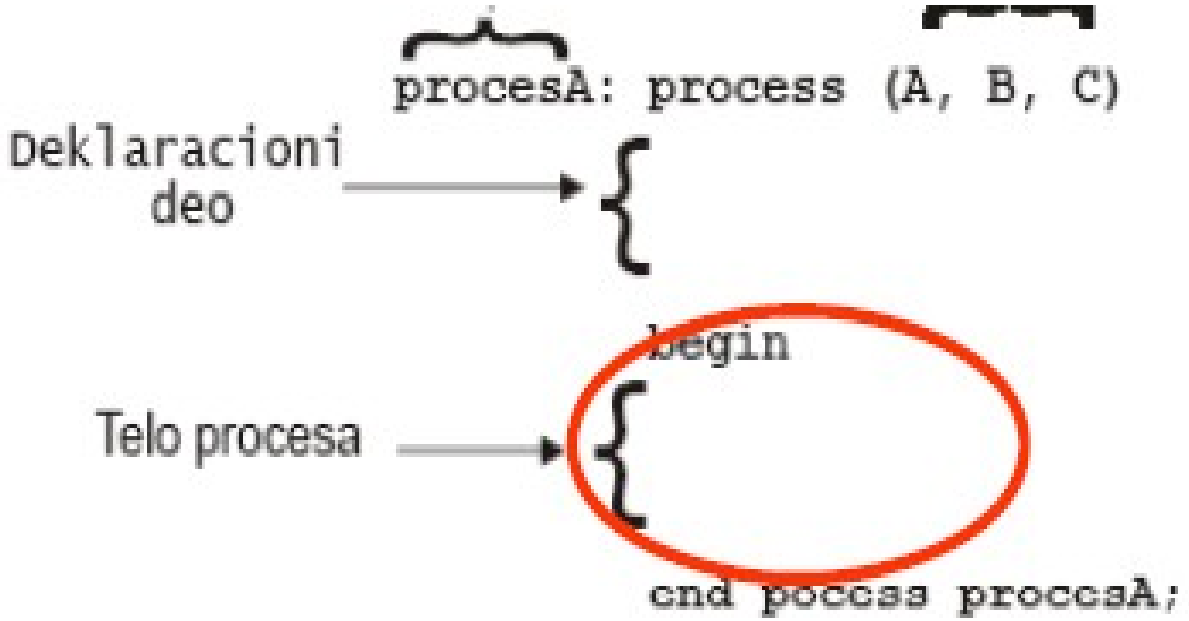
VHDL proces

Proces, kao i arhitektura, sadrži polje za deklarisanje elemenata (npr. promenljivih) lokalnih za taj proces



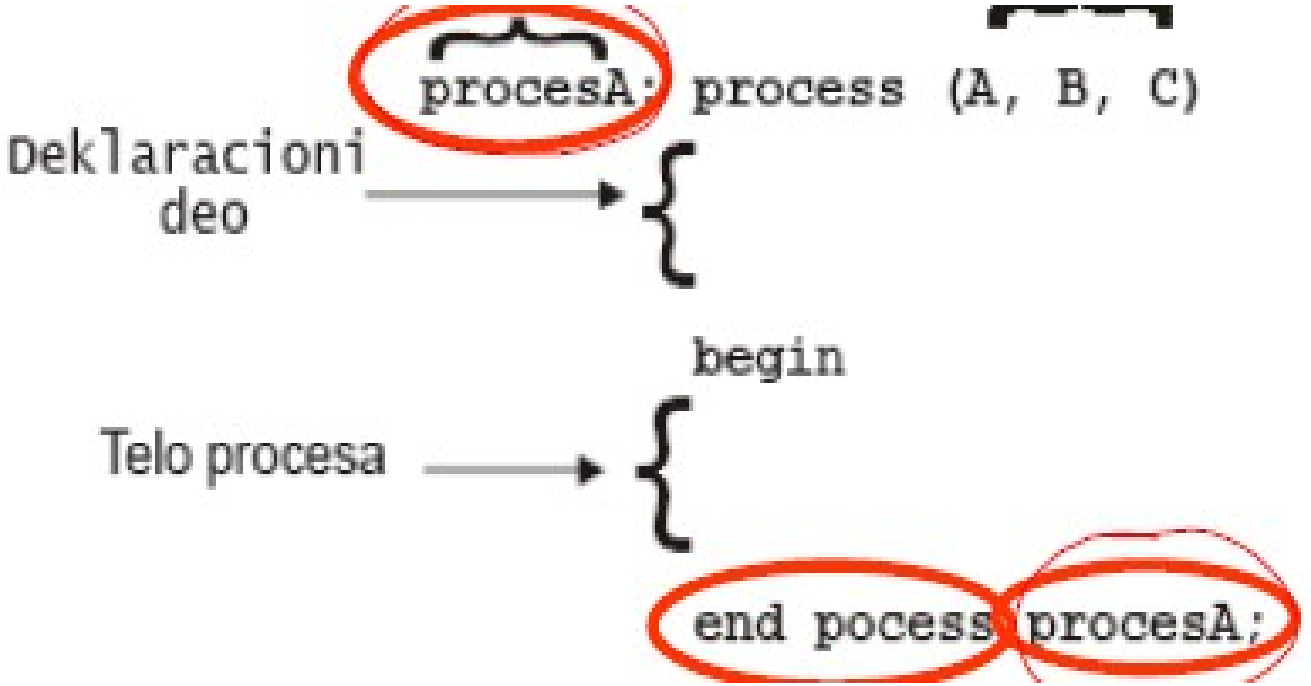
VHDL proces

Centralno polje zauzima telo procesa u kome se navode naredbe po striktno sekvencijalnom redosledu.



VHDL proces

Opis procesa završava se sa **end process**.
Opciono može da se doda ime procesa što se uglavnom i **preporučuje - procesA.**



VHDL Stilovi opisa projekta

VHDL podržava opis projekata na:
- algoritamskom nivou i
- nivou logičkih jednačina.

S obzirom da je VHDL i razvijan sa ciljem da se efikasno opišu složena kola, rezultujući kôd karakteriše jezgrovitost koja proističe iz hijerarhijskog pristupa dekompoziciji projekta.

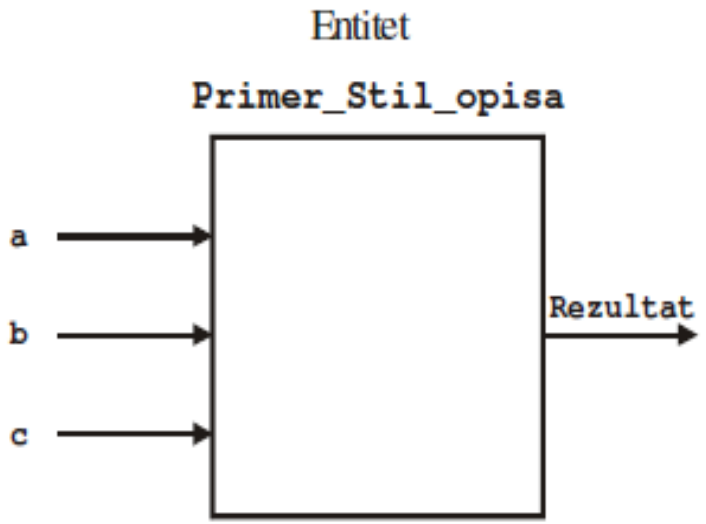
VHDL Stilovi opisa projekta

Postoje tri stila opisa projekta u VHDL-u:

- **strukturni opis (structural)**
- **opis ponašanja (behavioral)**
- **opis toka podataka (dataflow)**

VHDL Stilovi opisa projekta

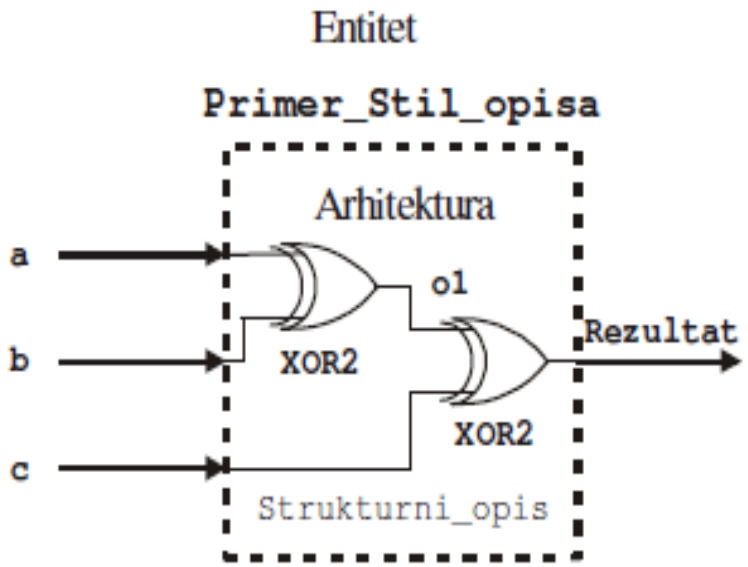
Primer stil opisa:




```
entity Primer_Stil_opisa is
 port
( a, b, c: in std_logic;
  Rezultat: out std_logic
);
end entity Primer_Stil_opisa;
```

VHDL Stilovi opisa projekta

Strukturni opis:




```
architecture Strukturni_opis of
             Primer_Stil_opisa is
  signal o1: std_logic; --deklaracija internih
         --signala

begin
  u1:   xor2 port map (  a => I1,    -- )
        b => I2,      -- )
        o1 => Y);       -- )
  u2:   xor2 port map ( o1 => I1,    -- } telo
        c => I2,      -- )
        Rezultat => Y);     -- )
end architecture Strukturni_opis;
```

VHDL Stilovi opisa projekta

Opis ponasanja:

```
architecture Opis_Ponasanja of Primer_Stil_opisa
is

begin

XOR_od_3: process (a, b, c)  -- davanje imena
             -- procesu nije neophodno,
            -- ali je korisno
begin
if((a XOR b XOR c) = '1') then
   Rezultat <= '1';
   else
   Rezultat <= '0';
end if;
end process XOR_od_3;

end architecture Opis_Ponasanja;
```

VHDL Stilovi opisa projekta

Tok podataka:

```
architecture Tok_podataka of Primer_Stil_opisa is
signal o1: std_logic;    -- deklaracija
          -- internog signala
begin
o1 <= I1 XOR I2;
I1 <= a;
I2 <= b;
Rezultat <= o1 XOR c;

end architecture Tok_podataka;
```