Osnovne jezičke konstrukcije u VHDL-u

Skelet jednostavnog VHDL programa

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

entity even_detector is
    port
        (
            a: in std_logic_vector(2 downto 0);
            even: out std_logic
        );
end even_detector;

architecture sop_arc of even_detector is
    signal p1, p2, p3, p4: std_logic;
begin
    even <= (p1 or p2) or (p3 or p4);
    p1 <= (not a(2)) and (not a(1)) and (not a(0));
    p2 <= (not a(2)) and  a(1) and a(0);
    p3 <= a(2) and (not a(1)) and a(0);
    p4 <= a(2) and a(1) and (not a(0));
end sop_arc;
```

VHDL program čini kolekcija dizajn jedinica (design unit). VHDL kod pogodan za sintezu mora sadržati bar dvije osnovne dizajn jedinice: deklaraciju entiteta i tijelo arhitekture koja pripada deklarisanom entitetu. Skelet tipičnog VHDL programa je prikazan kroz primjer.

Deklaracija entiteta opisuje spoljašnji interfejs, sadrži ime kola i imena i osnovne karakteristike ulaznih i izlaznih portova. U datom primjeru, deklaracija entiteta ukazuje da je ime kola **even_detector** i da kolo ima trobitni ulazni port **a** i jednobitni izlazni port **even.**

Deklaracija entiteta

```
entity entity_name is
    port
        (
            port_names: mode data_type;
            port_names: mode data_type;
            .   .   .
            port_names: mode data_type
        );
end entity_name;
```

Treba uočiti da se izostavlja **;** na kraju deklaracije posljednjeg porta.
Deklaracija porta se sastoji od imena porta, moda i tipa podatka. Modom se definiše smjer signala, koji može biti **in**, **out** i **inout** za bidirekcioni port.
Port deklarisan kao izlazni se ne može koristiti kao ulazni signal. Slijedi primjer.

Primjer: upotreba portova

```
-- pogrešna upotreba portova --

library ieee;
use ieee.std_logic_1164.all;

entity mode_demo is
    port
        (
            a, b: in std_logic;
            x, y: out std_logic
        );
end mode_demo;
architecture wrong_arc of mode_demo is
begin
    x <= a and b;
    y <= not x; -- !!! --
end wrong_arc;
```

```
-- moguće rješenje --

library ieee;
use ieee.std_logic_1164.all;

entity mode_demo is
    port
        (
            a, b: in std_logic;
            x, y: out std_logic
        );
end mode_demo;
architecture ok_arc of mode_demo is
    signal ab: std_logic;
begin
    ab <= a and b;
    x <= ab;
    y <= not ab;
end ok_arc;
```

Tijelo arhitekture specificira internu organizaciju ili operacije koje kolo obavlja. U VHDL-u je moguće definisati više arhitektura za jedan isti entitet koje se po potrebi vezuju za entitet prilikom simulacije ili sinteze.
Prvom linijom se definiše ime arhitekture odgovarajućeg entiteta. Tijelo arhitetkture može opciono da sadrži deklarativnu sekciju koja se sastoji od deklaracija objekata, kao što su signali i konstante, koji bi se koristili u opisu arhitekture.
Glavni segment tijela arhitekture se sastoji od izraza (concurrent statement) kojima se opisuje funkcionalnost kola. Svaki od izraza opisuje pojedinačni fizički element i arhitektura se može posmatrati kao kolekcija međusobno povezanih kola.

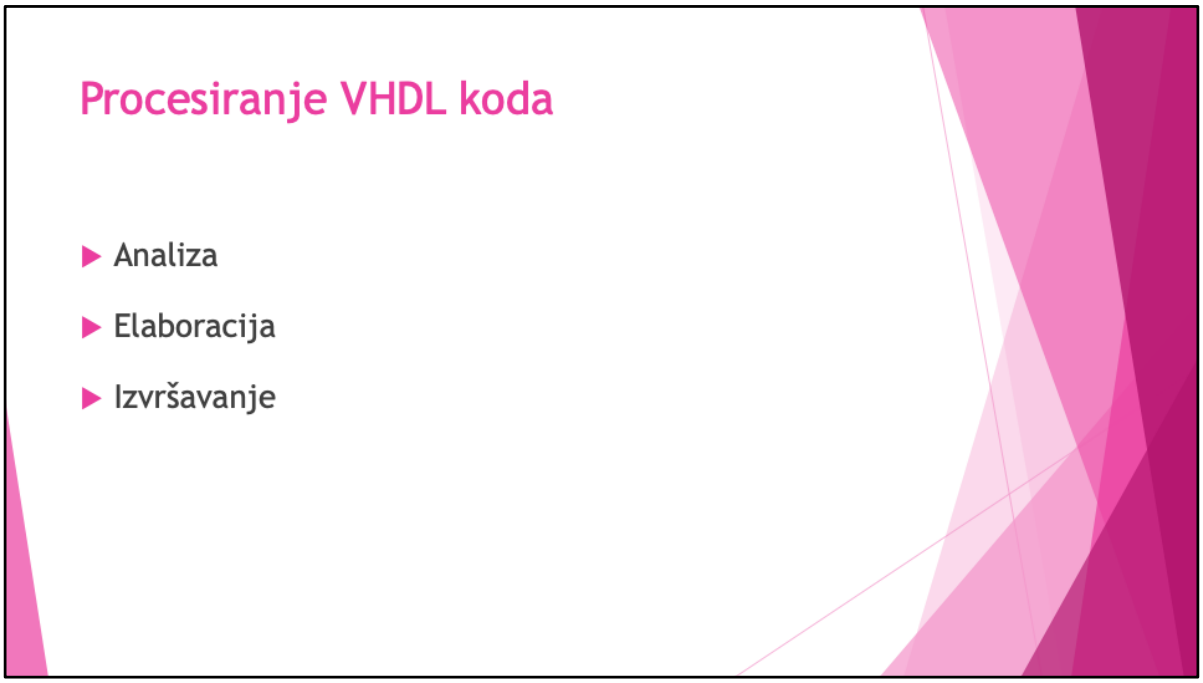
Dizajn jedinice (design units) su fundamentalni gradivni blokovi VHDL programa. Dok se program procesira, dijeli se na individualne cjeline (dizajn jedinice) i svaka jedinica se analizira nezavisno. Postoji 5 vrsta dizajn jedinica: deklaracija entiteta, tijelo arhitekture, deklaracija package-a, tijelo package-a, konfiguracija.
Package VHDL-a obično sadrži kolkeciju najčešće korišćenih objekata kao što su tipovi podataka, podprogrami i komponente, koje su neophodne od strane većine VHDL programa. Deklaracija package-a sadrži deklaraciju pomenutih objekata. Tijelo package-a sadrži implementaciju i kod potprograma.
U VHDL-u se za isti entitet može definisati vše tijela arhitektura. Konfiguracijom se specificira koja arhhitektura se povezuje sa određenim entitetom.
VHDL biblioteka je prostor u kome su smještene dizajn jedinice. Po default-u, dizajn jedinice na kojima se trenutno radi su u **work** biblioteci.

Kako bi se pojednostavila sinteza, IEEE je razvio nekoliko package-a, uključujući **std_logic_1164** i **numeric_std** , koji su definisani IEEE standardima 1164 i 1076.3.
Kako bi se koristili predefinisani package-i, mora se uključiti **library** i **use statement** prije deklaraciuje entiteta:

library ieee; -- uvođenje biblioteke ieee

use ieee.std_logic_1164.all; -- package std_logic_1164 je vidljiv

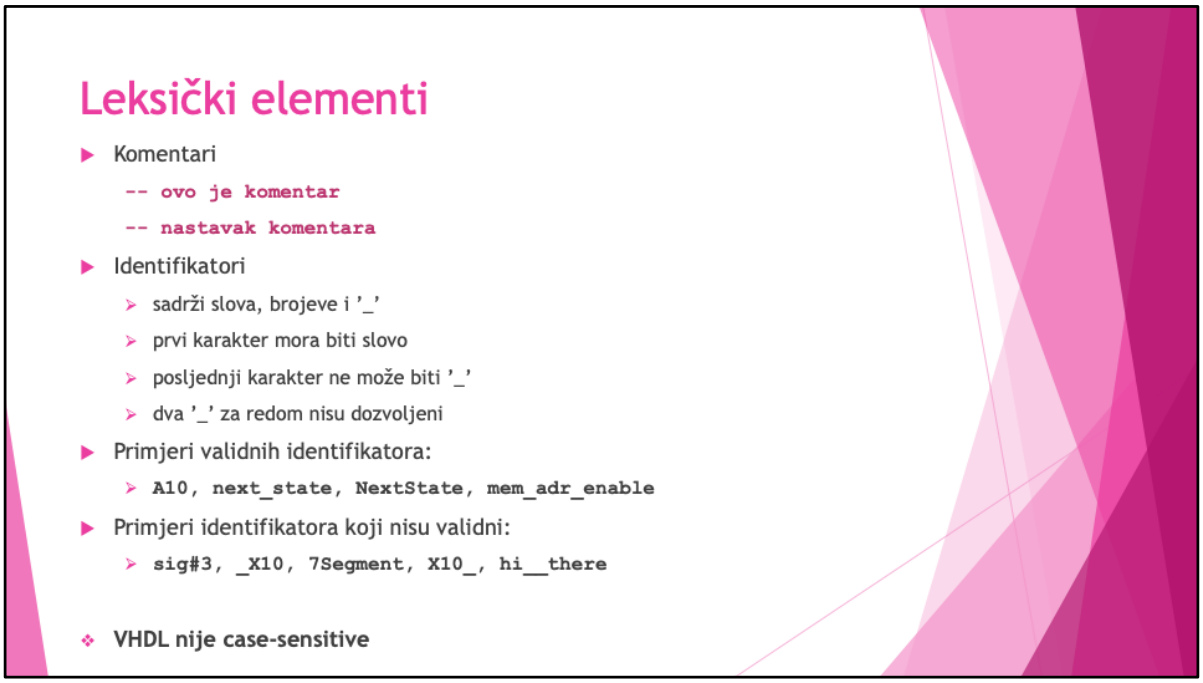
Prethodno je neophodno kako bi bilo moguće koristiti predefinisane tipove podataka kao što su **std_logic** i **std_logic_vector** iz std_logic_1164 package-a.



Za vrijeme faze analize, softver provjerava sintaksu i statičke semantičke greške VHDL koda. Analiza se sprovodi na osnovu dizajn jedinica. Ukoliko nema grešaka, softver prevodi kod dizajn jedinice u posrednu formu i čuva ga u odgovarajućoj biblioteci. VHDL fajl može da sadrži više dizajn jedinica, ali dizajn jedinica se ne može podijeliti u više fajlova.

Kompleksniji sistemi se obično hijerarhijski opisuju. U toku faze elaboracije, softver počinje od top-level entiteta, i vrši povezivanje arhitekture kako je opisano. Ukoliko postoje instancirane komponente, softver vrši zamjenu odgovarajućim arhitekturama. Proces se rekurzivno ponavlja dok je neophodno.

U toku faze izvršavanja, analiziran i elaboriran opis se obično prosljeđuje softveru za simulaciju ili sintezu. Ova faza podrazumijeva i fizičku implementaciju.



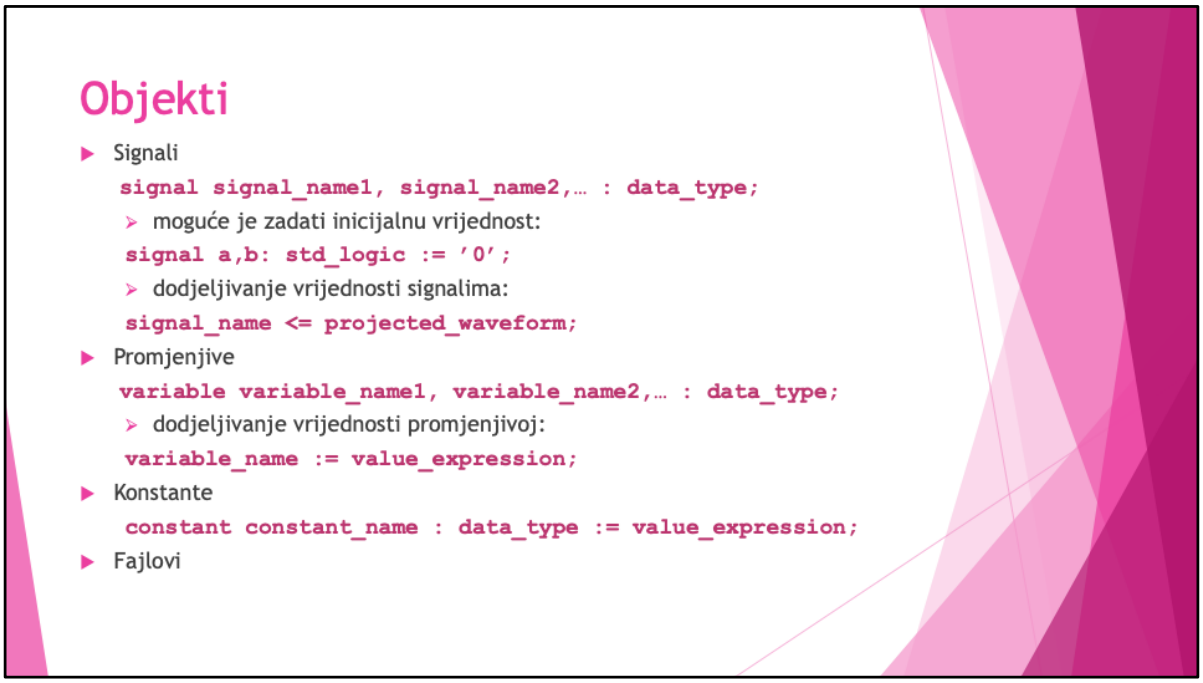
Leksički elementi predstavljaju osnovne sintatičke jedinice u VHDL programu.

Rezervisane riječi

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

Brojevi, karakteri, stringovi

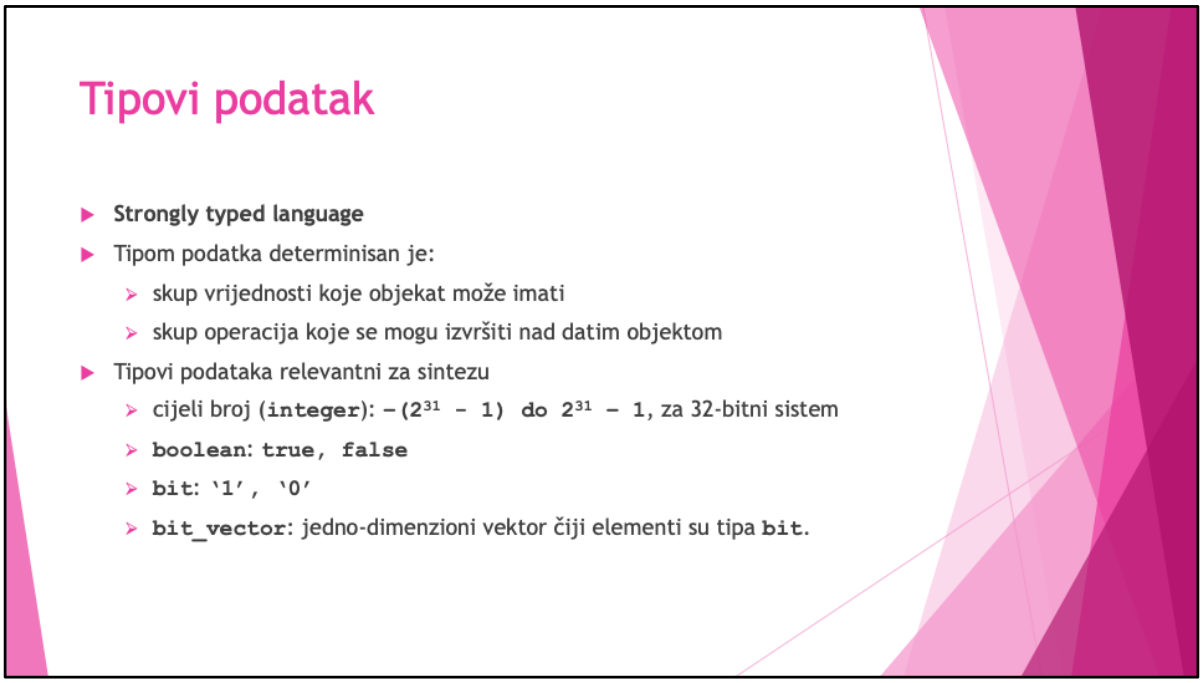
- Brojevi
  - cijeli
  - realni
  - u binarnom, dekadnom i heksadecimalnom brojnom sisemu
- Karkteri
  - navode se među jednostrukim navodnicima: ‘A’, ‘3’
- Stringovi
  - navode se među dvostrukim navodnicima: `“Hello”, “01001111”`



Sa aspekta sinteze, signal predstavlja žicu ili "žicu sa memorijom" (registar ili latch). Ulazni i izlazni portovi entiteta se takođe smatraju signalima.

Za varijablu ne postoji hardverska paralela. Može se posmatrati kao "simbolička memorijska lokacija" gdje se može smjestiti i modifikovati neka vrijednost. Varijabla se može deklarisati i koristiti samo u okviru procesa i lokalna je za taj proces. Njena osnovna uloga je apstraktni opis sistema. Može joj se dodijeliti samo vrijednost koja nije promjenjiva u vremenu, jer se za promjenjive ne vezuje bilo kakva informacija o vremenu. Dodjeljivanje := se označava kao immediate assignment (nema kašnjenja).

Konstante sadrže vrijednosti koje se ne mogu mijenjati.



Strongly typed language: Objektu može samo biti dodijeljena vrijednost istog tipa, i nad njim se mogu izvršiti samo operacije definisane za taj tip.
Kako osim logičke nule i jedinice postoji objektivna potreba za npr stanjem visoke impedanse, uvedeni su std_logic i std_logic_vector.

Operatori

Operator	Description	Data type of operand a	Data type of operand b	Data type of result
a ** b	exponentiation	integer	integer	integer
abs a	absolute value	integer		integer
not a	negation	boolean, bit, bit_vector		boolean, bit, bit_vector
a * b	multiplication	integer	integer	integer
a / b	division			
a mod b	modulo			
a rem b	remainder			
+ a	identity	integer		integer
- a	negation			
a + b	addition	integer	integer	integer
a - b	subtraction			
a & b	concatenation	1-D array, element	1-D array, element	1-D array

Operatori

a **sll** b	shift-left logical	bit_vector	integer	bit_vector
a **srl** b	shift-right logical			
a **sla** b	shift-left arithmetic			
a **srl** b	shift-right arithmetic			
a **rol** b	rotate left			
a **ror** b	rotate right			
a = b	equal to	any	same as a	boolean
a /= b	not equal to			
a < b	less than	scalar or 1-D array	same as a	boolean
a <= b	less than or equal to			
a > b	greater than			
a >= b	greater than or equal to			
a **and** b	and	boolean, bit, bit_vector	same as a	same as a
a **or** b	or			
a **xor** b	xor			
a **nand** b	nand			
a **nor** b	nor			
a **xnor** b	xnor			

Operatorti. Prioritet

Precedence	Operators
Highest	** abs not
	* / mod rem
	+ - (identity and negation)
	& + - (addition and subtraction)
	sll srl sla sra rol ror
	= /= < <= > >=
Lowest	and or nand nor xor xnor

Tipovi podataka u IEEE std_logic_1164 package-u

- **std_logic** može imati jednu od sljedećih vrijednosti:
  - '0' i '1' označavaju "forsiranje logičke nule" i "forsiranje ogičke jedinice"
  - 'Z' označava stanje visoke impednase, obično kod tro-statičkih bafera
  - 'L' i 'H' označavaju "slabu logičku nulu" i "slabu logičku jedinicu"
  - 'X' i 'W' označavaju "nepoznat" i "slab nepoznat". Signal ima vrjednost koja može biti interpretirana i kao logička nula i kao logička jedinica. Koristi se u simulacijama za pogrešan uslov.
  - 'U' označava neinicijalizovan. U simulacijama ukazuje da još uvijek nije dodijeljena vrijednost signalu ili promjenjivoj
  - '-' označava "don't care"

U okviru sinteze se koriste 0, 1 i Z stanja.

Tipovi podataka u IEEE std_logic_1164 package-u

- **std_logic_vector**

  ```
  signal a : std_logic_vector (7 downto 0);
  ```

- MSB

  ```
  a(7);
  ```

- LSB

  ```
  a(0);
  ```

- pristup

  ```
  a(7 downto 3);
  a(3);
  ```

Preklopljeni operatori

Overloaded operator	Data type of operand `a`	Data type of operand `b`	Data type of result
not `a`	`std_logic_vector` `std_logic`		same as `a`

`a` **and** `b`
`a` **or** `b`
`a` **xor** `b`
`a` **nand** `b`
`a` **nor** `b`
`a` **xnor** `b`
	`std_logic_vector`
	`std_logic`
		same as `a`	same as `a`

Konverzije tipova podataka

Function	Data type of operand a	Data type of result
to_bit(a)	std_logic	bit
to_stdulogic(a)	bit	std_logic
to_bitvector(a)	std_logic_vector	bit_vector
to_stdlogicvector(a)	bit_vector	std_logic_vector

Operatori za jedno-dimenzione nizove

- Relacioni operatori

```
"011" = "011", "011" > "010", "011">"00010", "0110">"011"
```

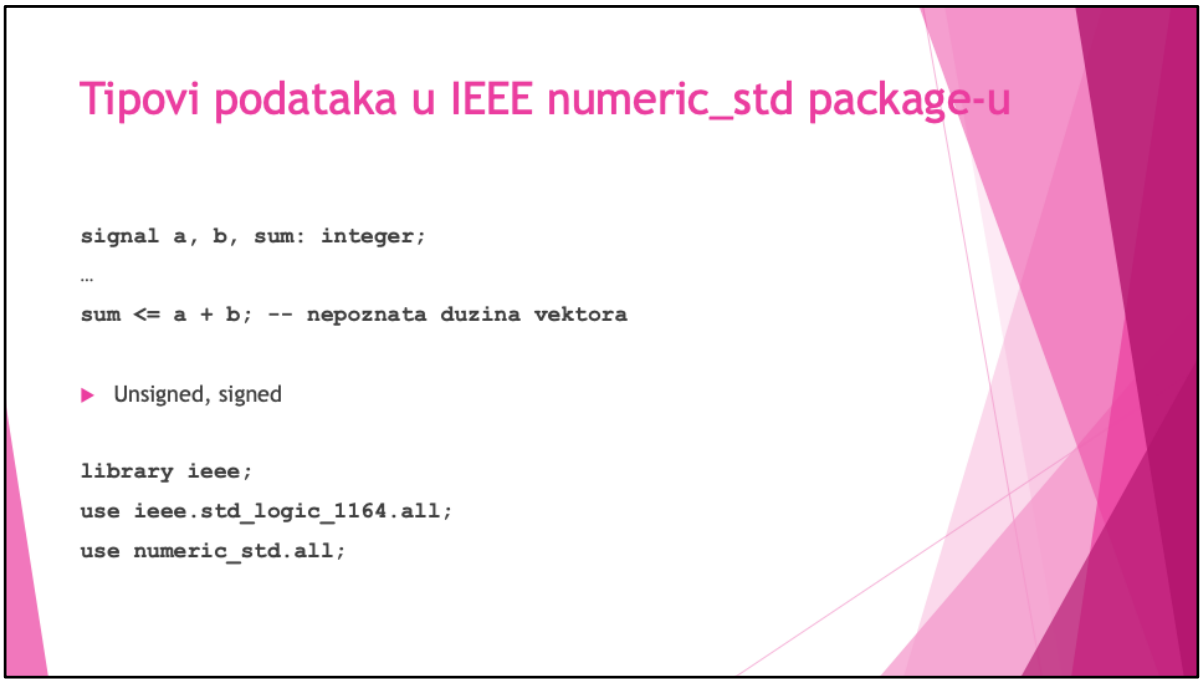
- Konkatenacija

```
y <= "00" & a(7 downto 0);
y <= a(7) & a(7) & a(7 downto 2);
y <= a(1 downto 0) & a(7 downto 2);
```

- Dodjela vrijednosti

```
a <= "10100000";
a <= ('1','0','1','0','0','0','0','0');
a <= (7=>'1', 6=>'0', 5=>'1', 0=>'0', 1=>'0', 3=>'0', 2=>'0', 4=>'0');
a <= (7|5=>'1', 6|4|3|2|1|0=>'0'); ili a <= (7|5=>'1', others=>'0');
a <= (others => '0');
```

Agreggate nije opartaor, to je konstrukcija u VHDL-u kojom se dodjeljuje neka vrjijednost objektu tipa niz.



Aritmetičke operacije nisu dozvoljene nad std_logic_vector tipom. Da bi se dva broja npr sabrala, moraju biti tipa integer. Koliko god jednostavno izgledao prikazani izraz, hardverska implementacoija nije jednostavna, naročito iz razloga što nije definisan opseg (broj bita) koje zauzimaju pojedini signali. Iako je simulacija trivijalna, za sintezu je značajna razlika ukoliko je u pitanju 8-bitni ili 32-bitni sabirač. Mnogo bolja alternative je koristiti niz nula i jedinica i interpretirati ga kao unsigned ili signed broj. Na taj način moguće je zadati dimenziju signala i ostavriti bolju kontrolu nad hardverom koji se konstruiše. IEEE numeric_std package je razvijen upravo u tu svrhu.

Preklopljeni operatori u IEEE numeric_std package-u

Overloaded operator	Description	Data type of operand a	Data type of operand b	Data type of result
abs a - a	absolute value negation	signed		signed
a * b a / b a **mod** b a **rem** b a + b a - b	arithmetic operation	unsigned unsigned, natural signed signed, integer	unsigned, natural unsigned signed, integer signed	unsigned unsigned signed signed
a = b a /= b a < b a <= b a > b a >= b	relational operation	unsigned unsigned, natural signed signed, integer	unsigned, natural unsigned signed, integer signed	boolean boolean boolean boolean

Funkcije u u IEEE numeric_std package-u

Function	Description	Data type of operand a	Data type of operand b	Data type of result
shift_left(a,b) shift_right(a,b) rotate_left(a,b) rotate_right(a,b)	shift left shift right rotate left rotate right	unsigned, signed	natural	same as a
resize(a,b)	resize array	unsigned, signed	natural	same as a
std_match(a,b)	compare '-'	unsigned, signed std_logic_vector, std_logic	same as a	boolean
to_integer(a)	data type conversion	unsigned, signed		integer
to_unsigned(a,b)		natural	natural	unsigned
to_signed(a,b)		integer	natural	signed

Konverzije tipova u IEEE numeric_std package-u

Table 3.8 Type conversions of numeric data types

Data type of a	To data type	Conversion function/type casting
unsigned, signed	std_logic_vector	std_logic_vector(a)
signed, std_logic_vector	unsigned	unsigned(a)
unsigned, std_logic_vector	signed	signed(a)
unsigned, signed	integer	to_integer(a)
natural	unsigned	to_unsigned(a, size)
integer	signed	to_signed(a, size)

Primjer sa konverzijom tipova

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

signal s1, s2, s3, s4, s5, s6: std_logic_vector(3 downto 0);
signal u1, u2, u3, u4, u5, u6, u7: unsigned(3 downto 0);
signal sg: signed(3 downto 0);
 . . .
u3 <= u2 + u1; -- ok.
u4 <= u2 + 1; -- ok.
u5 <= sg; -- not ok.
u6 <= 5; -- not ok.
u5 <= unsigned(sg); -- ok.
u6 <= to_unsigned (5,4) ; -- ok.
u7 <= sg + u1; -- not ok.
u7 <= unsigned(sg) + u1; -- ok, but be careful.
s3 <= u3; -- not ok.
s4 <= 5; -- not ok.
s3 <= std_logic_vector(u3); -- ok.
s4 <= std_logic_vector(to_unsigned(5,4)); -- ok.
s5 <= s2 + s1; -- not ok.
s5 <= std_logic_vector(unsigned(s2) + unsigned(s1)); -- ok.
```

Preporuke za sintezu

- koristiti **`std_logic`** i **`std_logic_vector`** umjesto **`bit`** i **`bit_vector`**
- koristiti **`numeric_std package`** i **`signed/unsigned`** tip za aritmetičke operacije
- koristiti opadajući opseg (**`downto`**) prilikom deklaracije tipova **`std_logic_vector, unsigned, signed`**
- koristiti zagrade za definisanje prioriteta izvršavanja operacija
- ne koristiti korisničke tipove ukoliko nije neophodno
- ne koristiti := prilikom dodjele inicijalne vrijednosti signal
- koristiti operande jednake dužine prilikom relacionih operacija