Osnovne jezičke konstrukcije u VHDL-u

- **Leksički elementi** (identifikatori, komentari, rezervisane reči, brojevi, karakteri, stringovi)
- **Objekti** (signali, varijable, konstante)
- **Tipovi podataka** (integer, boolean, bit, bit_vector, std_logic, std_logic_vector, signed, unsigned)
- **Polja** (1D, 1Dx1D, 2D)
- **Atributi** (atributi vektora, atributi signala)

Identifikatori

- Predstavljaju imena objekata u VHDL-u (signala, promenljivih, entiteta, arhitektura itd.).

Pravila za pisanje identifikatora:

- Mogu sadržati samo slova, decimalne cifre i crtu za podvlačenje.
- Prvi karakter mora biti slovo.
- Poslednji karakter ne može biti crta za podvlačenje.
- Dve uzastopne crte za podvlačenje nisu dozvoljene.

Ispravno:

- **A11, sledece_stanje, NextState, addr_en**

Neispravno:

- **x#8, _x3, 5linija, a10_, ab__cd**

Identifikatori

- VHDL ne pravi razliku između malih i velikih slova, a to važi kako za identifikatore, tako i za službene reči.
- Tako *a* i *A*, *ni_kolo* i *NI_Kolo*, kao i *Port*, *port* i *PORT* znače isto.

Komentari

- Komentar počinje sa dve crtice “--” posle kojih sledi tekst komentara.
- Komentari se koriste radi dokumentacije i nemaju uticaja na procesiranje kod. (Prilikom kompajliranja, celokupan tekst počev od “--” pa do kraja tekuće linije se ignoriše.)

Rezervisane (službene) reči VHDL-a

- Imaju posebno značenje u VHDL-u i ne mogu se koristiti kao identifikatori:

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

Brojevi

- Celi: 0, 123456 i 98E7 (= $98 \cdot 10^7$)
- Realni: 0.0, 1.2345 ili 6.82E6 (= $6.82 \cdot 10^6$)
- Dozvoljeno je korišćenje crte za podvlačenje: 12_3456 isto što i 123456
- Osim kao decimalni, brojevi se mogu predstaviti i u drugim brojnim osnovama.
- Na primer, 45 se može napisati kao **2**#101101# u osnovi 2, odnosno **16**#2D# u osnovi 16 (vrednost na početku zapisa ukazuje na brojnu osnovu, dok se vrednost broja zapisuje između znakova #)

Karakteri i stringovi

- Karakteri se pišu pod jednostukim navodnicima: ‘A’, ‘a’, ‘7’
- Stringovi se pišu pod dvostrukim navodnicima: “Alo”, “1000111”
- 1 nije isto što i ‘1’ (1 je broj, a ‘1’ karakter).
- 2#10110010# i “10110010” nisu isto, (2#10110010# je broj, a “10110010” string).

Objekti

- Služe za reprezentaciju i čuvanje vrednosti podataka određenog tipa:
- Signali
- Varijable
- Konstante

Signali

- Za povezivanje komponenti i razmenu podataka između entiteta (kao električne veze u fizičkom kolu)
- Deklaracija signala (u deklarativnom delu arhitekture):

SIGNAL ime_signala,..., ime_signala : TIP_PODATAKA;

- Primer: Deklaracija tri signala tipa *std_logic*:

SIGNAL x, y, z : STD_LOGIC;

- Inicijalna (početna) vrednost signala, dozvoljeno u simulaciji ali **ne** i u sintezi:

SIGNAL x, y, z : STD_LOGIC := '0';

- Dodela vrednosti signalu:

ime_signala <= izraz;

Signali

- Treba naglasiti da je dodela vrednosti signalu uvek *odložena* u odnosu na trenutak izračunavanja izraza.
- Takođe, izraz s desne strane znaka “<=“, u svom najopštijem obliku, ne definiše samo pojedinačnu vrednost, već kompletan talasni oblik signala, tj. niz vrednosti koje će se u odgovarajućim vremenskim trenucima dodeljivati signalu.

Varijable

- Za čuvanje lokalnih podataka u procesu ili proceduri (odgovaraju promenljivama iz programskih jezika)
- Deklaracija (u deklarativnoj sekciji procesa):

VARIABLE ime_var,..., ime_var : TIP_PODATAKA;

- Dodela vrednosti varijabli, dozvoljeno u simulaciji ali **ne** i u sintezi:

ime_varijable := izraz;

- Koriste se za čuvanje lokalnih podataka u sekvencijalnim sekcijama VHDL koda (procesima i potprogramima).
- Ne postoji neka precizna fizička interpretacija varijabli, a svoju osnovnu primenu nalaze u apstraktnom modeliranju.
- Za razliku od signala, ažuriranje vrednosti varijable je trenutno, odnosno u momentu izvršenja naredbe dodele.

Konstante

- Sadrže nepromenljivu vrednost.
- Može se deklarisati u entitetu, arhitekturi ili paketu:

CONSTANT ime_konstante : TIP_PODATAKA := izraz;

- Primer:

CONSTANT M : INTEGER := 32;

CONSTANT N : INTEGER := M/4;

```
1  ARCHITECTURE arch OF parity IS
2  SIGNAL x : STD_LOGIC;
3  BEGIN
4  . . .
5  p := '0';
6  FOR i IN 7 DOWNTO 0 LOOP
7  P := p AND a(i);
8  END LOOP;
9  . . .
```

```
1  ARCHITECTURE arch OF parity IS
2  SIGNAL x : STD_LOGIC;
3  CONSTANT N : INTEGER := 7;
4  BEGIN
5  . . .
6  p := '0';
7  FOR i IN N DOWNTO 0 LOOP
8  P := p AND a(i);
9  END LOOP;
```

- opis postaje razumljiviji
- naknadna promena vrednosti ovog parametra zahtevala bi samo modifikaciju vrednosti konstante, ali ne i bilo kakvu intervenciju u kodu tela arhitekture

Tipovi podataka

- Tip podataka definiše skup vrednosti i skup operacija koje se mogu primeniti nad podacima datog tipa.
- VHDL je ***strogo*** *tipiziran* jezik: operacije su legalne samo ako su tipovi operanada usklađeni

Podela tipova podataka:

- **Predefinisani**: ugrađeni u jeziku (*bit*, *bit_vector, boolean, integer*, ...)
- **Standardni**: dostupni kroz standardizovane pakete (*std_logic, std_logic_vector, unsigned, signed, ...*)
- **Korisnički** - uvodi ih projektant
- Pojedini tipovi podataka se mogu koristiti samo u simulaciji, dok se drugi mogu koristiti i u sintezi.

BIT i BIT_VECTOR

- Definiše dvonivovsku logiku
- Dozvoljene vrednosti: ‘0’ i ‘1’
- BIT_VECTOR je vektorska varijanta tipa BIT: definiše niz (vektor) bitova
- **SIGNAL x : BIT;** -- deklariše x kao jednobitni signal tipa BIT
- **SIGNAL y : BIT_VECTOR(3 DOWNTO 0);** -- y je 4-bitni vektor. (3 DOWNTO 0) definiše opseg indeksa bitova u vektoru i njihov poredak (u ovom slučaju opadajući)
- **SIGNAL w : BIT_VECTOR(0 TO 7);** -- w je 8-bitni vektor. (0 TO 7) definiše opseg indeksa bitova u vektoru i njihov poredak (u ovom slučaju rastući).
- **SIGNAL z : BIT := ‘1’ ;** -- z je jednobitni signal inicijalne vrednost ‘1’.

Skalari i vektori

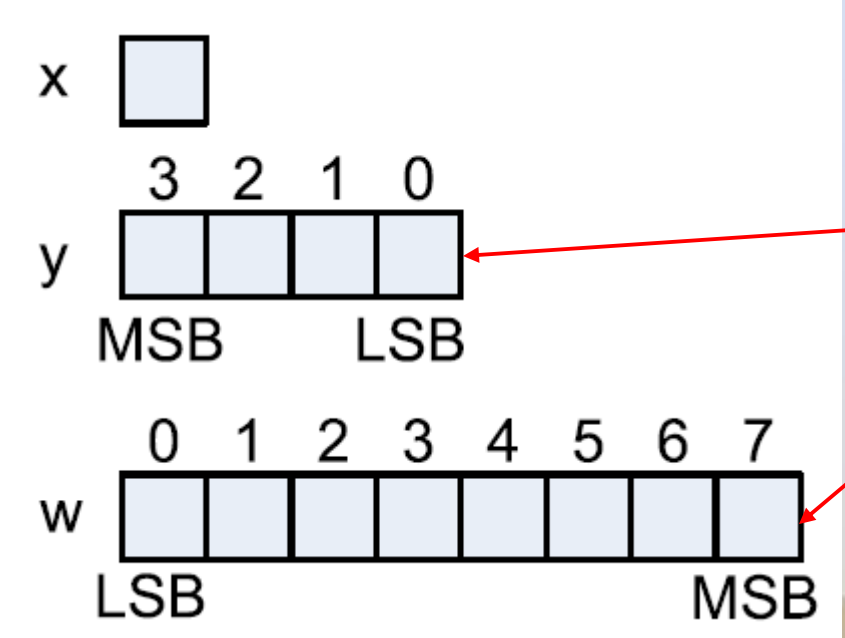
- Skalar

SIGNAL x : BIT;

- Vektor

SIGNAL y : BIT_VECTOR (3 DOWNTO 0);

SIGNAL w : BIT_VECTOR (0 TO 7);



Definiše dužinu vektora, opseg i poredak indeksa

Skalari i vektori

- **x <= ‘1’; --** jednobitnom signalu x dodeljuje se vrednost ‘1’, koriste se jednostruki znaci navoda ‘ ‘.
- **y <=“0111”; --** 4-bitnom signalu dodeljuje se vrednost “0111” (MSB=‘0’, binarna vrednost 7). Vektori se pišu pod dvostrukim navodnicima (“ “). Elementi vektora: y(3) , y(2), y(1), y(0) (koriste se male zagrade).
- **x <= y(2) ; --** skalaru x dodelje element vektora y sa indeksom 2. x = ?
- **w <=“01110001”; --** 8-bitnom signalu w dodeljuje vrednost 01110001(MSB=‘1’, binarna vredost 142).

BOOLEAN

- Definiše logičke vrednosti **true** (tačno) i **false** (netačno).
- Tip *boolean* nije isto što i tip *bit*:

```
IF (a) THEN ... -- a je tipa boolean
IF (a='1') THEN ... -- a je tipa bit
```

INTEGER

- 32-bitni celi (*integer*) brojevi.
- Opseg dozvoljenih vrednosti:

–(2^{31-1}) do 2^{31-1}, tj.

-2,147,483,647 do +2,147,483,647.

- U upotrebi su i dva izvedena tipa podatka (tj. podtipa):

NATURAL - obuhvata nenegativne cele brojeve (uključuje 0),

POSITIVE pozitivne cele brojeve (bez 0).

Operatori

- **Operatori su pridruženi definiciji tipa podataka i mogu se primenjivati samo na objekte tog tipa.**

Operacija	Opis	Tip operanda a	Tip operanda b	Tip rezultata
Logičke operacije				
NOT a	negacija	boolean, bit, bit_vector		**boolean, bit, bit_vector**
a AND b	I	boolean, bit, bit_vector,	isti kao a	**isti kao a**
a OR b	ILI			
a XOR b	isključivo ILI			
a NOR b	NILI			
a NAND b	NI			
a XNOR b	isključivo NILI			
Aritmetičke operacije				
a + b	sabiranje	integer	integer	**integer**
a - b	oduzimanje			
a * b	množenje			
a / b	deljenje			
a ** b	stepenovanje			
a MOD b	moduo			
a REM b	ostatak			
ABS a	apsolutna vrednost	integer		**integer**
- a	negacija			
Operacije poređenja				
a = b	jednako	bilo koji	isti kao a	**boolean**
a /= b	različito			
a < b	manje	skalar ili 1-D polje	isti kako a	**boolean**
a > b	veće			
a <= b	manje ili jednako			
a >= b	veće ili jednako			
Operacije pomeranja				
a sll b	logičko pomeranje ulevo	bit_vector	integer	**bit_vector**
a srl b	logičko pomeranje udesno			
a sla b	aritmetičko pomeranje ulevo			
a sra b	aritmetičko pomeranje udesno			
a rol b	rotacija na levo			
a ror b	rotacija na desno			
Konkatenacija				
a & b	**konkatenacija (nadovezivanje)**	**1D polje, element**	**1D polje, element**	**1D polje**

Logički operatori

- Za logičke operacije:
- Definisani za sledeće predefinisane tipove podataka:

NOT, AND, OR, NAND, NOR, XOR, XNOR

- Definisani za sledeće predefinisane tipove podataka:

BIT, BOOLEAN, STD LOGIC

- I odgovarajuća 1D polja:

BIT_VECTOR, STD LOGIC VECTOR

- Operator NOT je unarni i ima viši prioritet od svih ostalih operatora.

Logičke operacije nad skalarima

SIGNAL a, b, c, y: STD_LOGIC;

. . .

y <= NOT a AND b; -- (a'b)

y <= NOT (a AND b); -- (ab)'

y <= a NAND b; -- (ab)'

- Operatori NAND i NOR nisu asocijativni. Zbog toga, sledeća sintaksa nije ispravna:

y <= a NAND b NAND c;

- Neophodne su zagrade:

y <= (a NAND b) NAND c;

Logičke operacije nad vektorima

- Izvršavaju se nezavisno nad svakim parom odgovarajućih bitova dva vektora.
- Vektori moraju biti iste dužine.

```
SIGNAL a, b, c: STD_LOGIC_VECTOR(7 DOWNTO 0);
. . .
a <= “00111010”;
b <= “10000011”;
c <= NOT a; -- c dobija vrednost “11000101”
c <= a OR b; -- c dobija vrednost “10111011”
c <= a XOR b; -- c dobija vrednost “10111001”
```

Aritmetički operatori i operatori poređenja

- Ograničenja:
- Sabiranje, oduzimnje i množenje – **bez ograničenja**
- Deljenje – **samo ako je delilac stepen dvojke** (2^k)
- Stepenovanje – samo za definisanje vrednosti konstante
- MOD, REM i ABS **nisu dozvoljeni** u kodu za sintezu

+	Sabiranje
–	Oduzimanje
*	Množenje
/	Deljenje
**	Stepenovanje
MOD	Deljenje po modulu
REM	Ostatak deljenja
ABS	Apsolutna vrednost

=	Jednako
/=	Različito
<	Manje
>	Veće
<=	Manje ili jednako
>=	Veće ili jednako

Operatori pomeranja

- Pri pomeranju gubi se onoliko bitova za koliko pozicija se vektor pomera.
- **sll** - logičko pomeranje ulevo (u pozicije sa desne strane ‘0’)

BIT_VECTOR *x* = “01001”

y <= x sll 2; -- y <= “00100”

- **srl** - logičko pomeranje udesno (u pozicije sa leve strane ‘0’)

y <= x srl 3; -- y <= “00001”

- **U opštem slučaju, vrednost za popunjavanje je krajnja leva vrednost iz definicije odgovarajućeg tipa. BIT = {‘0’,’1’}**
- **sla** - aritmetičko pomeranje ulevo (pozicije sa desne strane popunjavaju se krajnjim desnim bitom vektora koji se pomera)

y <= x sla 2; -- y <= “00111”

- **sra** - aritmetičko pomeranje udesno (pozicije sa leve strane popunjavaju se krajnjim levim bitom vektora koji se pomera)

y <= x sra 2; -- y <= “?”

Operatori rotiranja

- Pri rotiranju bitovi se ne gube, jer se oni koji bi inače “ispali” iz vektora vraćaju na suprotan kraj vektora.

BIT_VECTOR *x* = “11001”

- **rol -** rotacija na levo

y <= x rol 2; -- y <= “01110”

- **ror -** rotacija na desno

y <= x ror 2; -- y <= “00111”

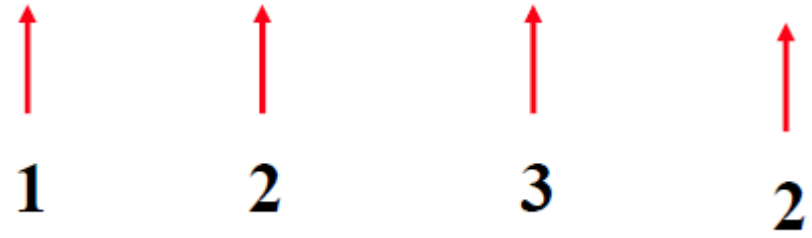
- **U kodu za sintezu operatore pomeranja dozvoljeno je koristiti samo nad operandima tipa BIT_VECTOR.**

Prioritet operatora

Prioritet	Operator
Najviši	** ABS NOT
	* / MOD REM
	- (negacija)
↓	& + -
	sll srl sla sra rol ror
	= /= < <= > >=
Najniži	and or nand nor xor xnor

Prioritet operatora

```
a + b > c OR a < d
```



Isto kao:

```
((a + b) > c) OR (a < d)
```

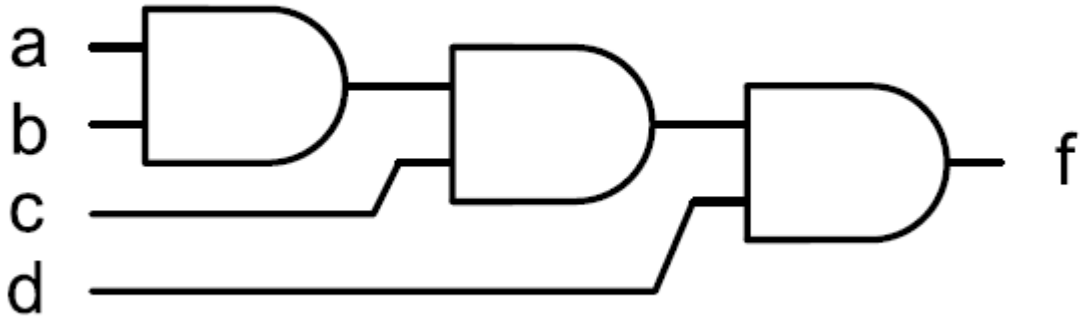
```
a + b + c + d
```

Isto kao:

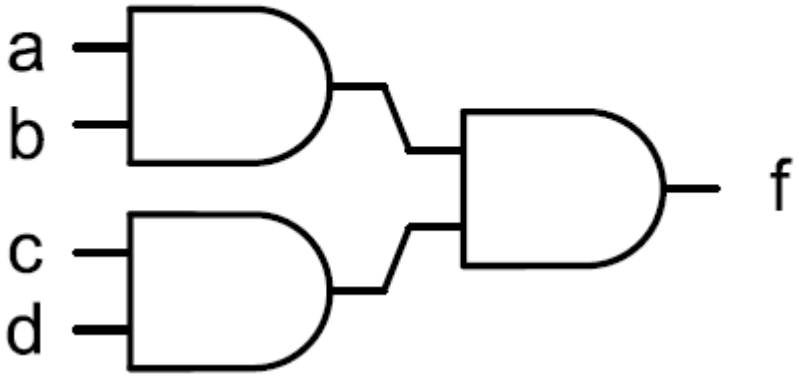
```
(((a + b) + c) + d)
```

Efekat zagrada

f = a AND b AND c AND d



f = (a AND b) AND (c AND d)



Tipovi podataka iz paketa std_logic_1164

- STD_LOGIC – skalarni tip
- STD_LOGIC_VECTOR – vektorski tip
- Uopštenje tipova BIT i BIT_VECTOR
- **Namenjeni sintezi**

STD_LOGIC

- Definiše 8-nivovski logički sistem. Dozvoljene vrednosti za sintezu

′X′	Nepoznata vrednost	
′0′	Nizak logički nivo	Sintetiše se kao logička ′0′
′1′	Visok logički nivo	Sintetiše se kao logička ′1′
′Z′	Visoka impedansa	Sintetiše se kao tro-statički bafer
′W′	″Slaba″ nepoznata vrednost	
′L′	′″Slab″ nizak nivo	
′H′	′″Slab″ visok nivo	
′-′	Proizvoljna vrednost	

Automatsko razrešavanje konflikta

U jednom momentu na vezi može postojati samo jedan logički nivo, problem će se javiti onda kada drajveri veze generišu različite logičke vrednosti - ova pojava se naziva *konfliktom* na magistrali.

Dozvoljeno u sintezi

	X	**0**	**1**	**Z**	**W**	**L**	**H**	**-**
X	X	X	X	X	X	X	X	X
0	X	0	X	0	0	0	0	X
1	X	X	1	1	1	1	1	X
Z	X	0	1	Z	W	L	H	X
W	X	0	1	W	W	W	W	X
L	X	0	1	L	W	L	W	X
H	X	0	1	H	W	W	H	X
-	X	X	X	X	X	X	X	X

STD_LOGIC i STD_LOGIC_VECTOR

SIGNAL x : STD_LOGIC; -- deklariše x kao jednobitni (skalarni) signal tipa STD_LOGIC

SIGNAL y : STD_LOGIC_VECTOR(3 DOWNTO 0) := “0001”; -- deklariše y kao 4-bitni vektor i dodeljuje mu inicijalnu vrednost “0001”

-- Inicijalna vrednost je opciona, a dodeljuje se operatorom “:=“

-- Krajnji levi bit je bit najveće težine (MSB).

Operacija nad tipom STD_LOGIC

- **Dozvoljeni su logički, ali ne i aritmetički operatori.**

Preklopljeni operator	**Tip operanda *a***	**Tip operanda *b***	**Tip rezultata**
NOT a	std_logic std_logic_vector		isti kao *a*
a **AND** b	std_logic std_logic_vector	isti kao *a*	isti kao *a*
a **OR** b			
a **XOR** b			
a **NOR** b			
a **XNOR** b			

Logičke operacije nad operandima tipa STD_LOGIC_VECTOR

- Logičke operacije nad vektorima izvršavaju se nezavisno nad svakim parom odgovarajućih bitova dva vektora. Pri tom vektori moraju biti iste dužine.

```
1  SIGNAL a, b, c: STD_LOGIC_VECTOR(7 DOWNTO 0);
2  . . .
3  a <= "00111010";
4  b <= "10000011";
5  c <= NOT a; -- c dobija vrednost "11000101"
6  c <= a OR b; -- c dobija vrednost "10111011"
7  c <= a XOR b; -- c dobija vrednost "10111001"
8  --
```

Funkcije za konverziju iz paketa *std_logic_1164*

Funkcija	**Tip operanda a**	**Tip rezultata**
to_bit(a)	std_logic	bit
to_stdlogic(a)	bit	std_logic
to_bitvector(a)	std_logic_vector	bit_vector
to_stdlogicvector(a)	bit_vector	std_logic_vector

Funkcije za konverziju iz paketa *std_logic_1164*

Pretpostavimo da su **s1, s2, s3, b1, b2** signali definisani na sledeći način:

SIGNAL s1, s2, s3 : STD_LOGIC_VECTOR(7 DOWNTO 0);

SIGNAL b1, b2 : BIT_VECTOR(7 DOWNTO 0);

Sledeće naredbe dodele su neispravne zbog neusklađenosti tipova:

s1 <= b1; -- signalu tipa std_logic_vector se dodeljuje vrednost signala tipa bit_vector

b2 <= s1 AND s2; -- vrednost tipa std_logic_vector se dodeljuje signalu tipa bit_vector

s3 <= b1 OR s2; -- operacija OR nije definisana između vrednosti tipa bit_vector i std_logic_vector

Treba ovako:

s1 <= to_stdlogicvector(b1);

b2 <= to_bitvector(s1 AND s2);

s3 <= to stdlogicvector(b1) OR s2; ili

s3 <= to_stdlogicvector(b1 OR to_bitvector(s2));

Operacije nad vektorskim tipovima

- Relacije
- Konkatenacija
- Agregacija

Relacije (poređenja) nad vektorima

- Sledeći izrazi su tačni:

“101” = “101”,

“011” > “010”,

“011” > “00011”,

“0110” > “011”

- ‘1’ je veće od ‘0’
- Poređenje je s leva na desno, do prvog neslaganja
- Duži vektor je ″veći″, ako je ″kraći″ u potpunosti sadržan u dužem.

Konkatenacija

- Nadovezivanje (spajanje) manjih vektora, segmenata vektora ili skalara u jedan veći vektor.
- Operator konkatenacije je znak “&”.

y <= x & “10000000”; -- ako je *x* = ‘1’, tada *y* <= “11000000”

y <= “00” & a(7 DOWNTO 2); -- ekvivalentno pomeranju vektora *a* za dve pozicije udesno

y <= a(1 DOWNTO 0) & a(7 DOWNTO 2); -- ekvivalentno rotaciji vektora *a* dve pozicije udesno

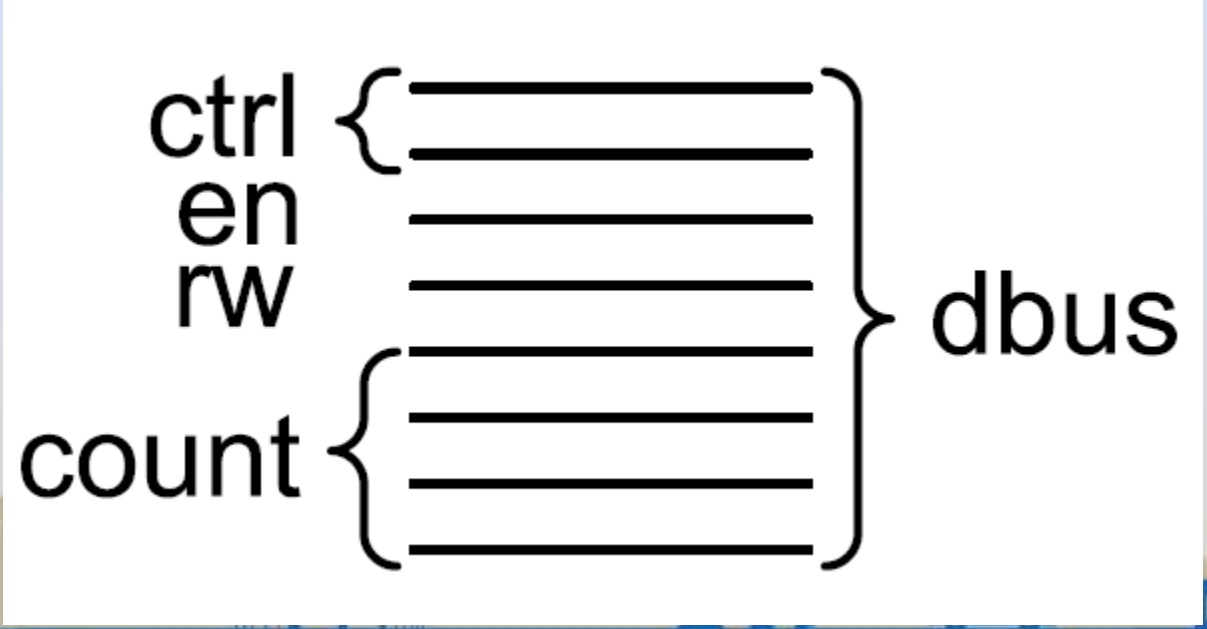
- Pomeranje pomoću konkatenacije

SIGNAL a, b: STD_LOGIC_VECTOR(7 DOWNTO 0);

a <= b(5 DOWNTO 0) & “00”; -- a <= b << 2 – pomeranje za dva mesta u levo

Konkatenacija

```
SIGNAL dbus: STD_LOGIC_VECTOR(0 TO 7);
SIGNAL ctrl: STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL en, rw: STD_LOGIC;
SIGNAL count: STD_LOGIC_VECTOR(0 TO 3);
dbus <= ctrl & en & rw & count;
```



Agregacija

SIGNAL w: STD_LOGIC_VECTOR(7 DOWNTO 0);

. . .

w <= “00001001”;

w <= (’0’,’0’, ’0’, ’0’, ’1’, ’0’, ’0’, ’1’);

w <=(7=>’0’,1=>’0’,5=>’0’,6=>’0’,0=>’1’,4=>’0’,3=>’1’,2=>’0’);

w <= (7|6|5|4|2|1 =>’0’, 3|0 =>’1’);

w <= (0=>’1’, 3=>’1’, OTHERS => ‘0’);

- Dodela svih nula vektoru:

w <= (OTHERS => ‘0’);

Tipovi podataka iz paketa *IEEE numeric_std*

- Uključuje se sa:

LIBRARY IEEE;

USE IEEE.NUMERIC_STD.ALL;

- Podrška za aritmetičke operacije
- Uvodi dva tipa podataka:

UNSIGNED – za neoznačene binarne brojeve

SIGNED – za označene binarne brojeve

SIGNAL a, b : SIGNED(7 DOWNTO 0);

SIGNAL x : UNSIGNED(7 DOWNTO 0);

```
SIGNAL a, b, r : INTEGER;
. . .
r <= a + b;
```

- Naredbu sabiranja iz prethodne naredbe teško je sintetizovati u hardver, zato što u kôdu nije eksplicitno naznačen opseg operanada *a* i *b*, pa nije ni poznato koliko bita je potrebno za njihovo predstavljanje.
- Informacija o ″dužini″ operanada, premda nebitna za simulaciju, od suštinske je važnosti za sintezu. Razlika u hardverskoj složenosti 8-bitnog i 32-bitnog sabirača je ogromna.
- Bolji i prirodniji pristup je da se za operande u aritmetičkim operacijama umesto tipa *integer* koriste nizovi (vektori) binarnih vrednosti ´0´ i ´1´ koji će se interpretirati kao neoznačeni ili označeni binarni brojevi.

Označeni i neoznačeni brojevi

Šta predstavlja *a* <= “1100” ?

Zavisi od konteksta (tipa signala/varijable kome se dodeljuje).

Ako je *a* je tipa **std_logic_vector,** tada je “1100” niz od 4 nezavisna bita.

Ako je *a* je tipa **unsigned,** tada je “1100” neoznačen binarni broj, vrednosti 12.

Ako je *a* je tipa **signed,** tada je “1100” označen binarni broj, vrednosti – 4.

Operacije nad UNSIGNED i SIGNED

Operacija	Opis	Tip operanda a	Tip operanda b	Tip rezultata
Aritmetičke operacije				
a + b	sabiranje	unsigned, natural signed, integer	unsigned, natural signed, integer	**unsigned signed**
a - b	oduzimanje			
a * b	množenje			
a / b	deljenje			
a MOD b	moduo			
a REM b	ostatak			
ABS a	apsolutna vrednost	signed		**signed**
- a	negacija			
Operacije poređenja				
a = b	jednako	**unsigned, natural signed, integer**	**unsigned, natural signed, integer**	**boolean**
a /= b	različito			
a < b	manje			
a > b	veće			
a <= b	manje ili jednako			
a >= b	**veće ili jednako**			

Operacije nad UNSIGNED i SIGNED

SIGNAL au, bu, cu, du, eu : UNSIGNED(7 DOWNTO 0);

SIGNAL as, bs, cs, ds : SIGNED(7 DOWNTO 0);

. . .

au <= bu + cu; -- *unsigned* i *unsigned*

du <= cu + 1; -- *unsigned* i *natural*

eu <= (3 + au + bu) - cu; -- *unsigned* i *natural*

as <= bs + cs; -- *signed* i *signed*

ds <= bs – 1; -- *signed* i *integer*

Dozvoljene i nedozvoljene operacije

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;
. . .
SIGNAL a, b, c: SIGNED(7 DOWNTO 0);
SIGNAL x, y, z: STD_LOGIC_VECTOR(7 DOWNTO 0);
. . .
c <= a + b;
-- ispravno, aritmetičke operacije su dozvoljene nad tipom signed
c <= a AND b;
-- neispravno, logičke operacije nisu dozvoljene nad tipom signed
z <= x + y;
-- neispravno, aritmetičke operacije nisu dozvoljene nad tipom std_logic_vector
z <= x AND y;
-- ispravno, logičke operacije su dozvoljene u tipu std_logic_vector
```

Relacije nad UNSIGNED/SIGNED

- Da li važi “011” > “1000” ?
- Ako su:
- Std_logic_vector: NETAČNO
- Unsigned: NETAČNO “011” je 3, a “1000” je 8
- Signed: TAČNO “011” je 3, a “1000” je -8

Funkcije za konverziju iz paketa *numeric_std*

Iz tipa	U tip	Funkcija za konverziju / eksplicitna konverzija
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)

- Treba zapaziti da ne postoji funkcija za konverziju iz tipa *std_logic_vector* u tip *integer* i obrnuto. Razlog za to je jednostavan - objekti tipa *std_logic_vector* se ne mogu interpertirati kao brojevi.

Eksplicitna konverzija tipa

- *Za konverziju podataka između tipova std_logic_vector, unsigned i signed*

SIGNAL u1, u2 : UNSIGNED(7 DOWNTO 0);

SIGNAL v1, v2 : STD_LOGIC_VECTOR(7 DOWNTO 0);

. . .

u1<= UNSIGNED(v1); -- konvertuje std_logic_vector u unsigned

v2<= STD_LOGIC_VECTOR(u1); -- konvertuje unsigned u std_logic_vector

SIGNAL u: UNSIGNED(7 DOWNTO 0);

u <= 5; -- neispravno, neusklađeni tipovi

Treba ovako:

u <= TO_UNSIGNED(5, 8);

Vrednost koja se konvertuje

Broj bitova u rezultujućoj binarnoj vrednosti

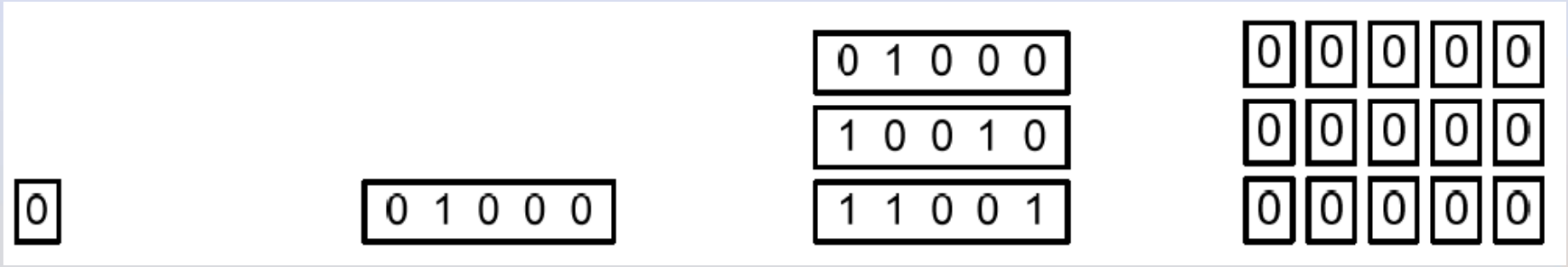
Korisnički tipovi i podtipovi podataka

- Mogućnost da projektant definiše svoje sopstvene tipove podataka. (celobrojni i nabrojivi)
- Podtip (*subtype*) je tip sa uvedenim ograničenjem.
- Ograničenje definiše podskup vrednosti osnovnog tipa koje su pridružene podtipu. Podtip nasleđuje sve operacije osnovnog tipa.

Polja

Kolekcije objekata istog tipa:

- Jednodimenziona (1D ili vektori)
- 1D x 1D
- Dvodimeniziona (2D ili matrice)
- Polja viših dimenzija (npr. 3D ili 2D x 2D) se ređe koriste
- i obično nisu podržana u alatima za sintezu



Polja

- Svi predefinisani tipovi podataka, kao i tipovi podataka iz paketa *std_logic_1164* i *numeric_std* su ili skalari ili vektori (tj. jednodimenziona polja skalara).
- Sledeći predefinisani tipovi se mogu **sintetizovati**:
- Skalari: *bit, std_logic* i *boolean*.
- Vektori: *bit_vector*, *std_logic_vector*, *integer*, *signed* i *unsigned*.
- Ne postoje predefinisana 2D ili 1D x 1D polja. Ako su neophodna, takva polja mogu biti definisana od strane projektanta:

TYPE ime_tipa IS ARRAY (opseg_indeksa) OF tip;

Polja (1D x 1D)

- Konstruišemo polje od 4 vektora dužine 8 bita. Pojedinačne vektore nazvaćemo *vrsta*, a celokupnu strukturu *matrica*.
- Usvojićemo da je bit najveće težine (MSB) svakog vektora njegov krajnji levi bit i da je prva vrsta matrice vrsta sa indeksom nula 0.

TYPE vrsta IS ARRAY (7 DOWNTO 0) OF STD_LOGIC; -- 1D polje

TYPE matrica IS ARRAY (0 TO 3) OF vrsta; -- 1D x 1D polje

SIGNAL x: matrica; -- 1D x 1D **signal**

- Isto što i:

TYPE matrica IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);

Polja (2D)

- Polje definisano sledećom linijom koda je “pravo” dvodimenzionalno polje, u potpunosti zasnovano na skalarima, a ne na vektorima, kao u prethodnom primeru:

TYPE matrica2D **IS ARRAY** (0 **TO** 3, 7 **DOWNTO** 0) **OF STD_LOGIC;**

Skalarni tip

Dvodimenzionalni indeks

Polja u naredbama dodele

```
1  TYPE vrsta IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;
2  TYPE polje1 IS ARRAY (0 TO 3) OF vrsta;
3  TYPE polje2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
4  TYPE polje3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;
5  SIGNAL x: vrsta;
6  SIGNAL y: polje1;
7  SIGNAL v: polje2;
8  SIGNAL w: polje3;
9  -- Ispravne skalarne naredbe dodele: ------------------------------
10 -- Sledeće skalarne (jednobitne) naredbe dodele su ispravne,zato što
11 -- je osnovni (skalarni) tip svih signala (x,y,v,w) std_logic
12 x(0) <= y(1)(2); -- uočimo dva para zagrada (y je 1Dx1D)
13 x(1) <= v(2)(3); -- dva para zagrada (v je 1Dx1D)
14 x(2) <= w(2,1); -- jedan par zagrada (w je 2D)
15 y(1)(1)<= x(6);
16 y(2)(0)<= v(0)(0);
17 y(0)(0)<= w(3,3);
18 w(1,1) <= x(7);
19 w(3,0) <= v(0)(3);
20 -- Vektorske naredbe dodele: ---------------------------------------
21 x <= y(0); --ispravno, isti tip podataka: vrsta
22 x <= v(1); --neispravno, neusklađeni tipovi: vrsta x
23 --std-logic-vector)
24 x <= w(2); --neispravno (pogrešno indeksiranje, w je 2D)
25 x <= w(2, 2 DOWNTO 0); --neispravno, neusklađeni tipovi
26 v(0) <= w(2, 2 DOWNTO 0); --neispravno, neusklađeni tipovi:
27 --std_logic_vector v i std_logic w
28 v(0) <= w(2); --neispravno, w je 2D
29 y(1) <= v(3); --neispravno, neusklađeni tipovi
30 y(1)(7 DOWNTO 3)<= x(4 DOWNTO 0); -- ispravno, isti tip i veličina
31 v(1)(7 DOWNTO 3)<= v(2)(4 DOWNTO 0); -- ok, isti tip i veličina
32 w(1,5 DOWNTO 1) <= v(2)(4 DOWNTO 0); -- neispravno, neuskl. tipovi
```

Atributi

- Dodatne informacije pridružene tipovima podataka, signalima i drugim objektima deklarisanim u VHDL kodu:
- Atributi vektora: sadrže informacije (vrednosti) koje se odnose na vektor
- Atributi signala: služe za nadgledanje signala

Primer 1

d´LOW	vraća najmanji indeks vektora d
d´HIGH	vraća naveći indeks vektora d
d´LEFT	vraća krajnji levi indeks vektora d
d´RIGHT	vraća krajnji desni indeks vektora d
d´LENGTH	vraća veličinu (dužinu) vektora d
d´RANGE	vraća opseg vektora d
d´REVERSE_RANGE	vraća inverzni opseg vektora d

SIGNAL d : STD_LOGIC_VECTOR(7 DOWNTO 0);

```
d´LOW = 0          d´LENGTH = 8
d´HIGH = 7         d´RANGE = (7 DOWNTO 0)
d´LEFT = 7         d´REVERSE_RANGE = (0 TO 7)
d´RIGHT = 0
```

Primer 2

SIGNAL d : STD_LOGIC_VECTOR(7 DOWNTO 0);

Za signal d sledeće četiri LOOP naredbe su ekvivalentne (brojač *i* uzima redom sve vrednosti definisane pridruženim opsegom):

FOR i IN RANGE (0 TO 7) LOOP . . .

FOR i IN d´RANGE LOOP . . .

FOR i IN RANGE (d´LOW TO d´HIGH) LOOP . . .

FOR i IN RANGE (0 TO d´LENGTH-1) LOOP . . .

Atributi signala

Dozvoljeni u sintezi

`s´EVENT`	vraća *true* ako se na signalu *s* desio događaj
`s´STABLE`	vraća *true* ako se na signalu *s* nije desio događaj
`s´ACTIVE`	vraća *true* ako je *s* = ´1´
`s´QUIET<vreme>`	vraća *true* ako se u navedenom vremenu na signalu *s* nije desio događaj
`s´LAST_EVENT`	vraća vreme proteklo od poslednjeg događaja na signalu *s*
`s´LAST_ACTIVE`	vraća proteklo vreme od kad je signal *s* poslednji put imao vrednost ´1´
`s´LAST_VALUE`	vraća vrednost signala *s* neposredno pre poslednjeg događaja

Primer 3

Rastuća ivica signala clk:

IF (clk´EVENT AND clk=‘1’) ... -- atribut EVENT u IF naredbi

IF (NOT clk´STABLE AND clk=‘1’) ... -- atribut STABLE u IF naredbi

WAIT UNTIL (clk´EVENT AND clk=‘1’); -- atribut EVENT u WAIT naredbi